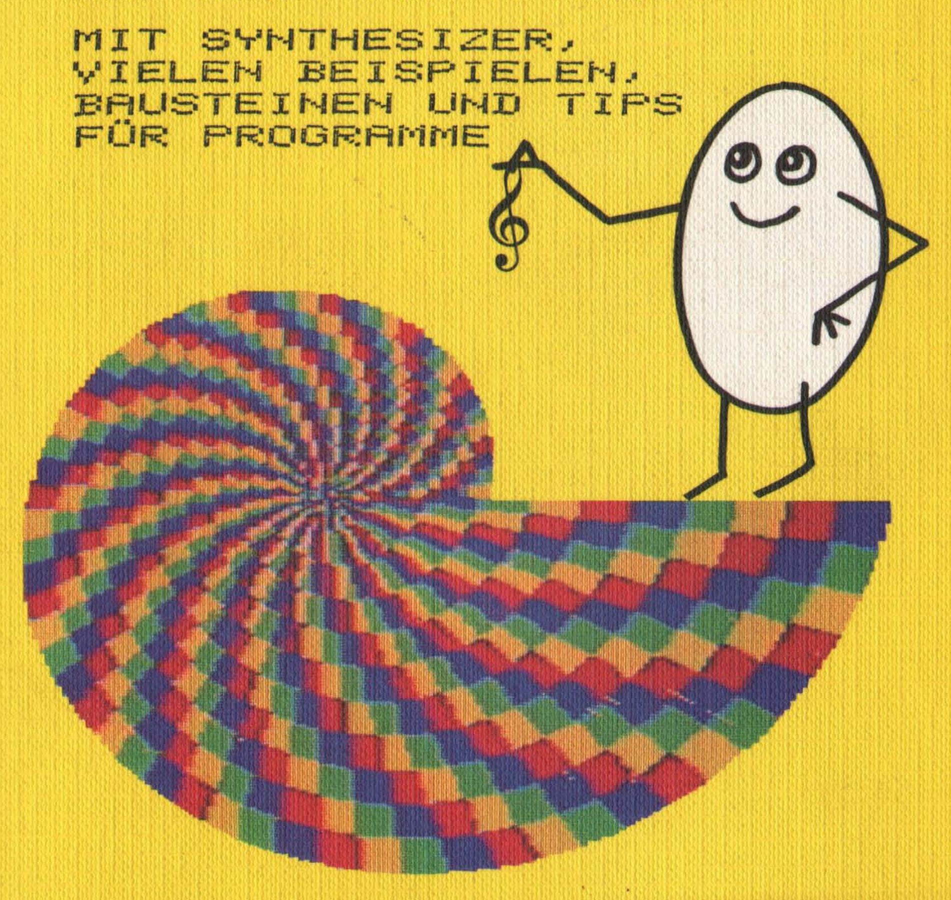
NORBERT TREITZ

BESSER PROGRAMMIEREN MIT DEM C 64

DAS BUCH ZUM HANDBUCH.

MIT SYNTHESIZER, VIELEN BEISPIELEN, BAUSTEINEN UND TIPS FÜR PROGRAMME



Hagemann

N O R B E R T   T R E I T Z

BESSER PROGRAMMIEREN MIT DEM C 64

DAS BUCH ZUM HANDBUCH.

MIT SYNTHESIZER,
VIELEN BEISPIELEN,
BAUSTEINEN UND TIPS
FÜR PROGRAMME

Hagemann

Hinweise auf weitere Arbeiten, Bücher und auf Programme des Autors finden Sie auf den Seiten 252 bis 256.

Impressum

Besser programmieren mit dem C64

Das Buch zum Handbuch

Autor: Dr. Norbert Treitz

Herstellung und Vertrieb:
HADÜ-Hagemann, Lehrmittel- und Verlagsgesellschaft mbH,
Düsseldorf
Printed in Germany ISBN 3-544-53002-3 Best.-Nr. 053002

Hinweis

In diesem Buch ist an mehreren Stellen von den Druckern VC 1515 und VC 1525 die Rede, auf die Programmbeispiele und der COPY-Befehl von SIMON's BASIC zugeschnitten sind. Das Nachfolgemodell dieser Drucker ist der Commodore MPS 8Ø1. Andere Drucker (wie der VC 1526 z.B.) sind zwar in anderer Hinsicht besser, eignen sich aber nicht so gut für die Feingrafik.

I N H A L T

Danksagung

Der Autor und der Verlag sind den folgenden Herren zu großem Dank verpflichtet, die das Manuskript auf Fehler und Unstimmigkeiten durchgesehen und manche Anregungen gegeben haben (soweit nicht anders vermerkt, alle aus Duisburg): Rüdiger Bartels, Andreas Denner, Norbert Hartmann, Alexander Hielscher (Rheinfelden/Baden), Wolfgang Mexner, Jörg Meyer, Uwe Ohse, Frank Reinicke (Essen), Jörg Unger und Michael Vinschen.

Ohne ihre Mitarbeit gäbe es in diesem Buch wesentlich mehr Irrtümer und schwerverständliche Passagen !

N.T.

Danksagung

[illegible]

[illegible]

[illegible]

WAS SOLL DIESES BUCH ?

Eine Art Vorwort

Nachdem "Besser programmieren mit dem VC 20" rasch großen Anklang bei seinen Lesern und Benutzern gefunden hat, entstand bald die Nachfrage nach einem entsprechenden Buch über den Commodore 64, der ja offenbar auf dem Markt zu einem zweiten Volkscomputer wird. Beide Bücher sind in mancher Hinsicht Zwillinge: die Abschnitte über das Microsoft-BASIC sind weitgehend identisch. Die Speicherplatz-"akrobatik", die Musik und die Grafik sind dagegen sehr unterschiedlich, die Sprite-Grafik ist etwas völlig Neues, die Musik hinsichtlich ihrer Qualität eigentlich auch.

Um die Feingrafik übersichtlich und schnell betreiben zu können, ist eine BASIC-Erweiterung unbedingt zu empfehlen. Die Programmbeispiele dazu in diesem Buch benutzen das von Commodore vertriebene SIMON's BASIC, erkennbar an der Einschaltanweisung HIRES für diese Grafik. Auch ist für bestimmte Teile und Beispiele anderes Zubehör erforderlich: Joystick,Paddles, Lichtgriffel und Drucker (vorzugsweise die Typen VC 1525 und VC 1515). Aber auch ohne Zubehör können Sie den größten Teil des Buches sinnvoll nutzen.

Während sonst lange Eintipp-Programme vermieden werden (zugunsten von kurzen Beispielen, die Sie als Bausteine für eigene Entwicklungen nehmen können), macht der Abschnitt über die Musik eine Ausnahme: Sie finden zwei Fassungen (mit und ohne Benutzung der SIMON-Feingrafik) eines SYNTHESIZERs, der zur Demonstration der aksutischen Fähigkeiten und ihrer Ansteuerungen dient: Sie spielen auf den Tasten Melodien, schalten auf anderen Tasten zwischendurch Klangfarben usw.um und bekommen bei jedem Umschalten die Einstellwerte grafisch auf dem Bildschirm angezeigt, in der SIMON-Version sogar mit Abbildung der Hüllkurven. Wenn Sie den SYNTHESIZER lieber als Cassette oder Diskette fertig kaufen wollen, statt ihn selbst einzutippen, können Sie das auch: Sie erhalten ihn im Handel oder direkt beim Verlag Hagemann.

Obwohl SIMON oder der Drucker über weite Strecken als verfügbar vorausgesetzt werden, kann keine Rede davon sein, die Möglichkeiten, die sich daraus ergeben, auch nur annähernd vollständig zu beschreiben. Das Schwergewicht liegt dagegen (nicht zuletzt aufgrund von Vorlieben, Beruf und Bequemlichkeit des Autors) bei kurzen grafikbetonten Programmen. Von daher erklärt sich auch, daß die sehr wertvollen strukturbetonten Elemente aus SIMON's BASIC, die sich bei längeren Programmen auszahlen, hier keine große Rolle spielen.

Sie finden in diesem Buch:

- Anmerkungen und Tricks zu BASIC-Anweisungen und gewissen POKE-Adressen
- besseres Verständnis der Grafik (und damit einige zusätzliche Tricks)
- Programmbausteine und kurze Beispielprogramme
- Anregungen zu eigenen Programmen

Sie finden <u>nicht</u> oder kaum:

- Maschinenprogrammierung,
- fertige längere Programme,
- Anleitungen zu Hardware-Arbeiten
- vollständige Verzeichnisse über Speicherplätze.

Solche Abhandlungen gibt es bereits in anderen Büchern, ebenso wie es viele geräteneutrale Bücher über BASIC gibt. Hier ist stattdessen einiges zusammengestellt, was dem fehlt, der eigene Programme schreiben will und dazu noch einige Tricks gebrauchen kann. Für Anregungen, Verbesserungsvorschläge und Kritik sind Autor und Verlag jederzeit dankbar.

Wie sollten Sie dieses Buch nutzen ?

Es ist nicht zum alleinigen Lesen gedacht: Ein C 64 sollte schon bereit stehen, damit Sie alles auch ausprobieren können. Auf der anderen Seite ist es auch keine Sammlung von Programmen zum gedankenlosen Eintippen; das können Sie zwar auch hin und wieder mit den Beispielen machen, aber im Endeffekt sollten Sie fast alles, was Sie da tippen, auch im Laufe der Lektüre verstehen, so daß Sie es gezielt abwandeln können. Viele Beispiele sind bewußt mager gehalten, quasi als Skelette für ausgebaute Programme, die Sie dann selbst daraus entwickeln können.

Die Abschnitte des Buches sind nicht so angeordnet, daß man sie von vorne nach hinten lesen muß und dabei nur das benutzt, was weiter vorne schon dran war. Es kann durchaus sein, daß irgendwo etwas benutzt wird, was weiter hinten noch ausführlicher erklärt wird. Meistens sind das Dinge, die Sie schon aus den normalen Gebrauchsanweisungen zu den Geräten mehr oder weniger kennen; benutzen Sie das Register sehr ausgiebig, wenn Sie Näheres über eine Anweisung oder ein anderes Stichwort suchen.

Lassen Sie sich auch nicht abschrecken, wenn Ihnen etwas zu "speziell" vorkommt: Es kommen einige mathematische Beispiele und Tricks vor, aber es ist nicht schlimm, wenn Sie nicht jeden Trick auch wirklich durchschauen (wer durchschaut schon alles, was er benutzt ?!). Vielleicht steigen Sie aber beim zweiten Lesen etwas tiefer z.B. in die Binärzahlen und die logischen Verknüpfungen ein und genießen dann die Eleganz ihrer Anwendungen.

Die wichtigste Regel: Probieren Sie eigene Ideen aus; dieses Buch soll Ihnen Anregungen dazu geben, nicht mehr, aber auch nicht weniger !

WIE IST EIN BASIC-PROGRAMM ÜBERHAUPT AUFGEBAUT ?

Man könnte fast sagen: völlig beliebig. Etwas ernsthafter: Formal besteht ein BASIC-Programm aus Zeilen, deren jede mit einer Nummer beginnt und im übrigen nur zugelassene Zeichen enthält. Inhaltlich gibt es keine vorgeschriebenen Regeln für die Gliederung: Anfang und Ende müssen nicht besonders gekennzeichnet werden. Es gibt aber eine ganz wichtige Regel: Wir müssen den Computer so durch das Programm lenken, daß er alle Informationen, die er benötigt, bekommt, bevor er sie benutzen soll. Soll er zum Beispiel etwas mit X malnehmen, so muß er zu dem Zeitpunkt schon "wissen", welchen Wert X haben soll; es nützt nichts, wenn er es später zu lesen bekommt. Denken Sie sich also immer beim Programmieren ein Männchen, daß Ihre Anweisungen in der Reihenfolge der Zeilen mit Beachtung der Sprünge liest und ausführt; das Männchen hat ein sehr gutes Gedächtnis für alles schon Gelesene, ist aber kein Hellseher.Von den mathematischen Symbolen kennt es nur die in BASIC eingebauten, und es versteht auch nur genau die Schreibweise, die hier benutzt wird.

Jede Programmzeile darf auf dem C 64 bis zu zwei Bildschirmzeilen lang sein (bei der Eingabe; wenn Sie die Schlüsselwörter abgekürzt schreiben oder die Leertaste nach der Zeilennummer nicht eintippen, kann die Zeile nachher in der AusLISTung länger sein: das ist genau dann störend, wenn Sie sie noch ändern wollen). Sie beginnt mit einer Zeilennummer, nämlich einer ganzen Zahl zwischen Ø und 63999(beide eingeschlossen). Außer dieser Nummer enthält die Zeile eine oder mehrere Anweisungen, zwischen denen je ein Doppelpunkt steht. Die Zeilen werden im Rechner automatisch nach aufsteigenden Zeilennummern geordnet. Beim Lauf (RUN) werden sie auch in der gleichen Reihenfolge bearbeitet, sofern nicht durch Sprunganweisungen in ihnen etwas anderes vorgeschrieben ist. Weder die letzte Zeile der Ausführung noch die letzte Zeile der Liste werden besonders gekennzeichnet. Das Programm endet, wenn es auf END

oder STOP stößt oder die letzte Zeile bearbeitet hat. Im Textmodus erscheint dann READY mit einigen Zeilen Vorschub, was oft ein mit der Tastatur erzeugtes Bild zerstört; im SIMON-HIRES-Modus schaltet das Programm nach dem Ende in den Textmodus zurück. In solchen Fällen empfiehlt es sich, den Rechner etwa so in eine Endlosschleife laufen zu lassen: 63999 GOTO 63999.
Es ist durchaus zu empfehlen, ein Programm zu gliedern und insbesondere einen Vorbereitungsteil zusammenzustellen, in dem Dimensionierungen, INPUT-Abfragen und Wertzuweisungen von Konstanten in Variable (besonders wenn sie öfter vorkommen) enthalten sind. Über DATA muß hier etwas im Vorgriff gesagt werden, da hier das Prinzip verlassen wird, nach dem ein Programm mit Sprüngen abgearbeitet wird: Die Sprünge (mit GOTO oder GOSUB) sind auf die Reihenfolge der DATA unwirksam; beim Einlesen mit READ sucht der Computer also im Programmtext den nächsten Inhalt einer DATA-Anweisung, völlig unabhängig vom übrigen Programm. Es ist daher auch gleichgültig, ob die DATA vor oder nach den zugehörigen READ-Anweisungen stehen: In Programmen mit Sprüngen stellt man am besten alle DATA am Anfang oder am Ende zu einem Block zusammen, und zwar in der Reihenfolge, in der das Programm mit READ zugreift.

PLATZHALTER:

Variable

Unter Variablen kann man sich Speicherplätze vorstellen, die statt der Nummern Namen tragen; der Computer kümmert sich alleine darum, wo er sie unterbringt. Es gibt drei Sorten: String (=Text) und zwei Sorten von Zahlen, nämlich kurze und lange (vornehmer gesagt: 5-Byte-Zahlen mit Gleit-Dezimalpunkt und 2-Byte-Ganzzahlen). Nehmen wir zunächst die normalen (langen) Zahlen:

Sie werden auch als REAL-Zahlen, also reelle Zahlen bezeichnet, obwohl es beim Computer keine unendlich langen Dezimalzahlen geben kann. Sie erhalten die einfachsten Variablennamen: z.B. A AS B3 YX usw.; das erste Zeichen ist auf jeden Fall ein Buchstabe. Danach kann noch eine Ziffer oder ein zweiter Buchstabe folgen. Es ist in BASIC zulässig, die Namen noch länger zu machen, aber der C 64 kümmert sich nur um die ersten beiden Zeichen, er nimmt also z.B. SUMX und SUMY für die gleiche Variable. Außerdem muß man aufpassen, daß in den Namen keine BASIC-Schlüsselwörter vorkommen, wie TAN in STANDARD oder AND in ANDREAS. Es ist daher sehr zu empfehlen, nur zwei Zeichen für jede Variable zu nehmen. Da ist die Gefahr, ein Schlüsselwort zu verwenden, geringer: nicht zulässig sind die Zeichen: FN GO IF ON ST TI TO und bei den Stringvariablen TI$.

Eine Variable steht in einem BASIC-Programm stets stellvertretend für ihren augenblicklichen Wert (nämlich während der Programmausführung !). Ob A = B ist, hängt also oft von dem Zeitpunkt ab, in dem danach gefragt wird.

("Reelle") Zahlen werden nicht dezimal, sondern als Binärzahlen gespeichert, und zwar in 5 Bytes*: 7 bit für den Zweier-Exponenten und 1 für dessen Vorzeichen (nämlich von -128 bis 127), 1 bit für das Vorzeichen der Zahl selbst und die restlichen 31 bit für die Ziffernfolge (was etwa einer Genauigkeit von 10 Dezimalstellen entspricht). Der Zahlenbereich ist damit ca. auf $\pm 10^{38}$ begrenzt. Beim Überschreiten bricht das Programm mit Overflow-Meldung ab. Viel schlimmer ist es, daß Zahlen ca. zwischen $+10^{-38}$ und -10^{-38} einfach auf Ø gesetzt werden, ohne daß eine Fehlermeldung kommt: Treten solche Zahlen als Zwischenergebnisse auf, so verschwinden sie, ohne sich abzumelden !

** 1 Byte besteht aus 8 Bit, ein Bit ist die Information einer Binärziffer, d.h. die Entscheidung zwischen Ø und 1.*

BEHÄLTER FÜR TEXTE

String-Variable

Daß man "Computer" nicht mit "Rechner" übersetzen sollte, wird sehr deutlich, wenn man sieht, daß in BASIC die Verarbeitung von Textstücken ("Strings") fast genau so gut möglich ist wie die von Zahlen. Ein String kann bis zu 255 Zeichen lang sein (Sie können aber mit INPUT ein so langes String nicht an einem Stück eingeben). Die String-Variablen erhalten als Kennzeichen $ am Schluß, z.B. A$, M4$ oder bei indizierten String-Variablen (auf die wir noch kommen) BX$(J) usw. Es können auch Zahlen als Texte behandelt werden, aber der Computer kann dann nicht ohne weiteres mit ihnen rechnen.

ZAHLEN AM LAUFENDEN BAND

Indizierte Variable

Oft hat man Variable, die zusammengehören und mit denen dasselbe gemacht werden soll, z.B. bestimmte Angaben über 8 verschiedene Staaten. Man könnte nun die Einwohnerzahlen E1 bis E8 nennen und die Flächen F1 bis F8. Soll der Computer nun die Quotienten ausrechnen, so bräuchte man 8 einzelne Anweisungen: Q1=E1/F1 usw. Mit den indizierten (d.h. numerierten) Variablen geht das viel einfacher: Die Einwohnerzahlen seien E(1) bis E(8), die Flächen F(1) bis F(8). Jetzt kann der Computer selbst die Indices von 1 bis 8 einzeln drannehmen, etwa so:

```
10 FOR I=1 TO 8: INPUT E(I),F(I):NEXT
20 FOR I=1 TO 8: PRINT E(I)/F(I):NEXT
```

Das lohnt sich natürlich erst, wenn noch mehr mit diesen Zahlen geschehen soll. Die Indices dürfen auch Ø sein, aber nicht negativ. Man kann auch höhere ganze Zahlen verwenden, z.B. von 35 bis 43, aber man benötigt dann den Speicherplatz für die darunterliegenden Nummern mit.*

---

* *außerdem ist ab Index 11 die Anweisung DIM nötig.*

Damit sind wir bei einem wichtigen Thema: dem Speicherplatzbedarf der Variablen. Solange es sich um einzelne handelt, hält er sich in Grenzen, wenn aber ein Index bis 1ØØØ laufen soll, lohnt sich ein näherer Blick:

In indizierten Variablen benötigt der C 64 pro Zahl folgende Plätze:

5 Bytes für normale (lange = REAL) Zahlen,
2 Bytes für kurze (INTEGER) Zahlen, vgl. unten,
3 Bytes für Strings.

Soll also der Index J einer Variablen X(J) bis 700 laufen dürfen, so werden 5·700 Bytes = 3500 Bytes benötigt (es sind genaugenommen noch ein paar mehr, die aber nicht ins Gewicht fallen).

Daß der Computer bei den Strings mit 3 Bytes auskommt, obwohl die doch bis zu 255 Zeichen (also 255 Bytes) lang sein dürfen, verdanken wir einem pfiffigen Trick: Dieser bescheidene Platz wird benötigt, um die Stelle zu kennzeichnen, an der das String ohnehin schon steht (z.B. im Programmtext in einer DATA-Zeile): Dann genügen 2 Bytes für die Anfangsadresse (16 bit wegen der 2^{16} Adressen im C 64) und ein Byte für die Länge (die ja genau deswegen auf 255 begrenzt ist).

PLATZSPAREN:
INTEGER-Variable mit %

Wenn Sie nun der Meinung sind, daß Sie lieber viele kleine (oder abgerundete) Zahlen haben wollen als wenige ausführliche, können Sie die sogenannten "Integer-Variablen" verwenden, deren Kennzeichen* ein % am Schluß ist, z.B.

---

* *Das Symbol % hat nichts mit der normalen Bedeutung (Prozent) zu tun.*

AX% oder BV%( ) oder L%( ). Es lohnt sich jedoch nur bei indizierten Variablen und auch dann nur, wenn man viele Bytes einsparen kann (bei nichtindizierten gibt es überhaupt keine Einsparung).

Damit der Computer nun mit 2 Bytes auskommt, um sich eine Zahl zu merken, kann diese nur 2^{16} verschiedene Werte haben. Es ist festgelegt, daß dies die ganzen Zahlen von -32768 bis +32767 sein sollen (bekanntlich ist $2^{15}=32768$).

Wenn man in einem Programm sehr viele Zahlen speichern und verarbeiten will, kann es selbst beim C 64 nötig werden, statt der Fließpunktzahlen auf INTEGER-Zahlen auszuweichen, und daß dabei die tatsächlichen Zahlen über den genannten Bereich hinausgehen oder auch zwischen 0 und 1 liegen. In beiden Fällen müssen wir uns zu helfen wissen. Nehmen wir an, daß MA eine Zahl ist, die auf jeden Fall größer als der Betrag der größten (bzw. kleinsten negativen) in Frage kommenden Zahl ist. Wir teilen dann alle Zahlen durch MA und multiplizieren sie mit 2^{15} oder mit 30000. Jetzt haben wir sie in dem Bereich, den wir mit INTEGER-Variablen behandeln können,und speichern sie ab. Später müssen wir die Umrechnung natürlich wieder irgendwann rückgängig machen. Dieses Verfahren ist auch dann sinnvoll, wenn es sich etwa um Zahlen zwischen 0 und 100 handelt, die aber nicht von selbst ganzzahlig sind: Ohne die Umrechnung würden sie durch das Abrunden auf 1/100 der größten Zahl ungenau gemacht, mit unserem Trick aber nur auf 1/30000 der größten ungenau.

PLATZKARTENPFLICHT:

Dimensionierung mit DIM

Format: DIM v(n)
DIM v(n1,n2) usw.
DIM v1(n1),v2(n2) usw.

mit: Variablennamen v() v1() v2()
n n1 n2 positive ganze Zahlen als jeweils höchste Indices

Im Speicher liegen die Werte für indizierte Zahlenvariable bzw. die "Zeiger" für Stringvariable schön hintereinander; dazu muß der Computer wissen, wie viele es werden sollen. Um uns einerseits Arbeit zu ersparen und andererseits mit dem Speicherplatz nicht zu verschwenderisch umzugehen, räumt er jeder indizierten Variablen, die im Programm auftaucht, Plätze für die Indices Ø bis 1Ø ein. Falls wir mehr brauchen, müssen wir das vorher (!) anmelden, ebenso bei mehrfacher Indzierung (z.B. X(I,J)). Für diese Voranmeldung des benötigten Platzes gibt es die Anweisung DIM (von "Dimension"), die vor dem ersten Erscheinen der entsprechenden Variablen bearbeitet werden muß. Soll z.B. die Stringvariable A2$( ) mit den Indices bis 23 laufen, so heißt die Anweisung: DIM A2$(23), bei mehrfachindizierten ähnlich: DIM A%(3,5) kündigt an, daß der erste Index bis 3, der zweite bis 5 laufen darf. Beachten Sie bitte, daß es zunächst völlig gleichgültig ist, ob Sie den Index später I oder K3 oder sonstwie nennen: Bei der Dimensionierung geht es nur um die Zahl (auch sonst können Sie eine Variable, die Sie als A(I) einlesen, später als A(K) abfragen usw.).

Die Flexibilität von BASIC bewährt sich auch daran, daß die Dimensionen erst im laufenden Programm entschieden werden müssen, z.B. :

```
10 INPUT"ZAHL DER SPIELER ";N
20 DIM N$(N),P(N)
30 FOR I=1 TO N
40 PRINT"WIE HEISST SPIELER NR.";I
50 INPUT N$(I):NEXT
```

Hier werden also eine Namensliste und für jeden Spieler ein Punktkonto eröffnet.

Beachten Sie bitte, daß nur Variable dimensioniert werden können, die noch nicht vorher benutzt (oder gar ausdrücklich dimensioniert) worden sind.

GANZE TABELLEN

Mehrfache Indizierung

Die einfache Indizierung entspricht Zahlenkolonnen. Oft braucht man auch Zahlen, die logisch in eine Tabelle (Matrix) gehören oder sogar in eine Staffel von Tabellen. Das wird an einem einfachen Beispiel klar, bei dem es um die Wahlergebnisse von drei Parteien bei vier Wahlen geht:

```
10 DIM S(2,3)
20 FOR I=0 TO 2:FOR J=0 TO 3
30 PRINT"PARTEI NR."I+1;"WAHL NR.";J+1:INPUT S(I,J)
40 NEXT:NEXT
50 FOR I=0 TO 2:SU=0:FOR J=0 TO 3
60 SU=SU+S(I,J):NEXT
70 PRINT"DURCHSCHNITT PARTEI ";I+1;SU/4:NEXT
```

DER GERETTETE HI-SCORE

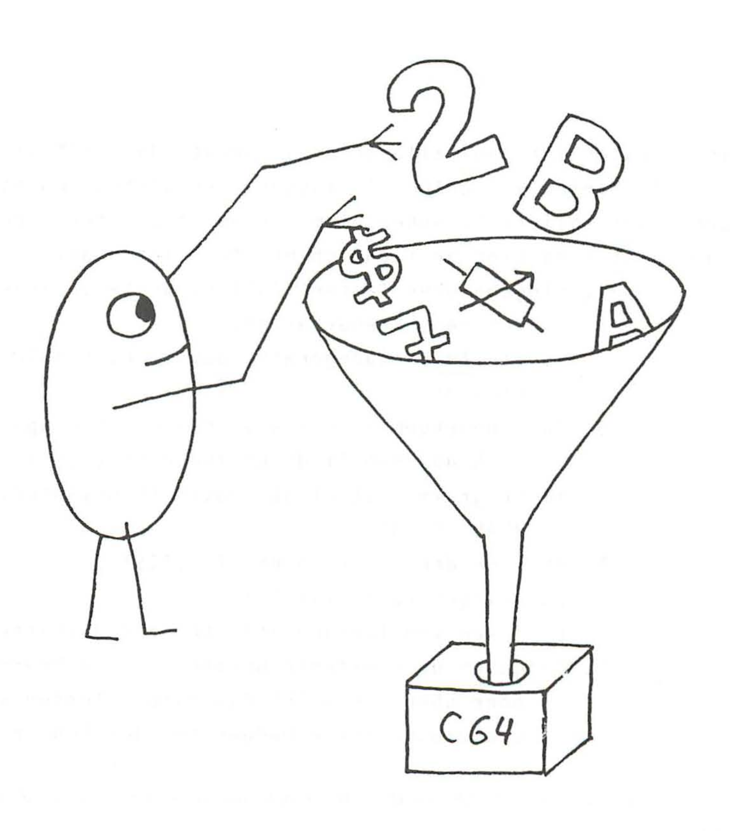
Daten vor RUN und CLR bewahren

Wenn Sie ein Programm neu starten (mit RUN, auch mit einem adressierten RUN, z.B. RUN 1ØØØ), werden alle Variablen gelöscht, d.h. auf die Zahl Ø bzw. auf das leere String "" gesetzt. Mit der Anweisung CLR (die man eigentlich kaum jemals benötigt) erreicht man dasselbe. Wollen Sie nun Daten aus einem Programmlauf, z.B. eine Maximalpunktzahl (Hi-Score) als Information in das nächste Spiel retten, so können Sie entweder das neue Spiel mit GOTO starten und alle Variablen, die gelöscht werden dürfen und müssen, durch ausdrückliche Wertzuweisungen löschen. Eine andere Möglichkeit besteht darin, Daten durch das Löschen hindurchzumogeln, indem man sie mit POKE in einem Speicherbereich ablegt, der nicht von CLR (und damit auch nicht beim Löschen durch RUN) erfaßt wird, z.B. im Bildschirmspeicher, oder noch besser in seiner "unsichtbaren" Verlängerung: der Bildschirm braucht nur 1ØØØ Plätze (von 1Ø24 bis 2Ø23), dahinter sind noch einige bis 2Ø47 frei, allerdings werden die Nummern 2Ø4Ø bis 2Ø47 für die Sprites benötigt.

Mit einer POKE-Anweisung können Sie ein Byte (also eine Zahl von Ø bis 255) oder auch nur eine Dezimalziffer ablegen, wenn Sie das übersichtlicher finden.

DIE HERKUNFT

DER DATEN

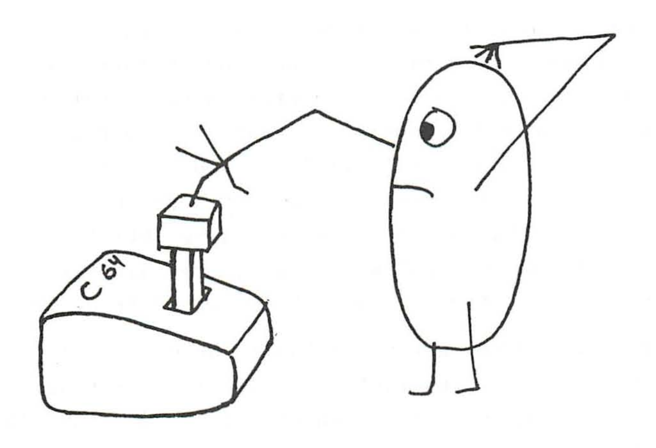


Eingabeverfahren und anderes

Ein Computer ist ganz allgemein ein Gerät, das mit Daten, die sich als Zahlen verschlüsseln lassen, irgendetwas macht und dann etwas abliefert. Woher nimmt er nun die Daten ? Der C 64 bietet da eine erstaunlich breite Palette an:

1. Eingabe über Tasten (Zahlen, Wörter, einzelne Zeichen, Steuertasten)
2. spezielle Eingabegeräte: Joystick, Paddles, Lichtgriffel
3. Zwischenergebnisse aus seinem eigenen Speicher, auch aus dem Bildschirmbereich (TEST)
4. im Programm festgelegte Daten (Konstanten, READ - DATA)
5. Abfrage der internen Uhr (TI,TI$),
6. Zufallsgenerator (RND())
7. Einlesen von Tonbandcassette oder Diskette
8. Eingaben über externe Geräte, die am User-Port oder über einen IEC-Bus angeschlossen sind und z.B. externe Meßgeräte oder Fühler ablesen.

In diesem Buch geht es hauptsächlich um die ersten 6 Möglichkeiten.



MIT 10 FINGERN BLIND ?

Die Tastatur - ganz übersichtlich

Der wichtigste Weg, etwas in den C 64 einzugeben, führt über die Tastatur. Wenn Sie mit Schreibmaschinen vertraut sind, müssen Sie auf Y und Z aufpassen: die sind hier vertauscht. Außerdem müssen Sie sehr genau zwischen dem Buchstaben O und der Ziffer Ø unterscheiden. Fast alle Tasten sind doppelt oder sogar dreifach belegt. Deshalb gibt es nicht nur den gewöhnlichen Umschalter (SHIFT), sondern auch noch einen zweiten (C=, auch Commodore-Taste genannt) und einen dritten: CTRL (control). Sie haben alle gemeinsam, daß man sie gedrückt halten muß, während man eine normale Taste anschlägt, deren zweite oder dritte Bedeutung man haben will (also nicht gleichzeitig anschlagen, weil dann der Umschalter möglicherweise etwas zu spät kommt; aber auch nicht zu früh loslassen). Die Zuordnung ist ziemlich

einfach: CTRL ist zum einen für die Farben, die auf den Ziffern-Tasten angesiedelt sind; leider stimmen die Ziffern auf den jeweils gleichen Tasten nicht mit den Code-Nummern der Farben überein, sondern sind um 1 versetzt: Schwarz hat die Nummer Ø, etwa in der HIRES-Anweisung, kann aber im Klartext einer PRINT-Anweisung mit Control-Taste und Taste 1 gewählt werden. Zum anderen wird CTRL auch für das Ein- und Ausschalten von REVERSE, zusammen mit den Tasten RVS ON bzw. RVS OFF, verwendet: bei REVERSE werden sozusagen die Farben von Papier und Tinte vertauscht (auch Negativschrift genannt).

Bei den dreifach belegten Tasten ist SHIFT für die rechts unten angeschriebenen Zeichen zuständig, die links von der linken SHIFT-Taste angebrachten "Commodore-Taste" C= für die links unten angeschriebenen und erreicht die Farben 8-15.

Drücken Sie auf SHIFT und C= zugleich, kommen Sie in den zweiten Zeichensatz, bei dem Sie ohne SHIFT Kleinbuchstaben schreiben und mit SHIFT Großbuchstaben, wie bei einer normalen Schreibmaschine. Die Grafikzeichen, die rechts unten an den Tasten stehen, fehlen in diesem Zeichensatz. Das Umschalten von einem Zeichensatz in den anderen wirkt sich auf den ganzen Bildschirm aus, also auch auf das schon Geschriebene, man kann also nicht beide zugleich im Bild haben. Beim Programmieren spielt es keine Rolle, welchen Sie verwenden; programmierte Umschaltungen macht man entweder mit PRINT CHR$(14) (in den zweiten Satz, also dem mit den Kleinbuchstaben) bzw. PRINT CHR$(142) (zurück zum ersten mit dem vollständigen Grafiksatz). Man kann das auch mit POKE bewerkstelligen: POKE 53272,21 schaltet auf den 1., POKE 53272,23 auf den zweiten Zeichensatz. In SIMON's BASIC geht das auch mit CSET Ø bzw. CSET 1 (innerhalb eines Programms kann man auch mit CSET 2 den Grafikbildschirm zurückholen, ohne ihn zu löschen).

Stehen auf der Oberseite einer Taste zwei Aufschriften, so gilt ganz allgemein die obere für SHIFT, die untere für die Taste allein (z.B. CLR und HOME).

Während die Wahl des Zeichensatzes auf den ganzen Bildschirm (auch rückwirkend !) Einfluß nimmt, gelten die Farben und das Ein- und Ausschalten von REVERSE für alles, was danach geschrieben wird (unabhängig davon, wohin es geschrieben wird), bis Sie etwas neues bestimmen. Wenn Sie allerdings ein Gänsefüßchen setzen, werden Farben oder REVERSE nicht gewechselt, sondern es erscheint stattdessen ein seltsames reverses Zeichen. Normalerweise ist das sehr sinnvoll: Beim Programmieren einer PRINT-Anweisung mit einem Klartext (z.B. PRINT "BITTE WARTEN !") kann man auf diese Weise Farbe oder RVS ON mitprogrammieren, wobei es sinnlos wäre, wenn nun die Anweisung selbst auch bunt erschiene. Nach dem Gänsefüßchen wird die Farbsteueranweisung also nicht ausgeführt, sondern stattdessen symbolisch vermerkt. Es gibt also für die Farben und für REVERSE einen direkten und einen indirekten (Programmier-) Modus: durch Tippen des Gänsefüßchens " schalten Sie zwischen beiden hin und her (wobei ein Löschen des Zeichens " auf dem Bildschirm mit DEL oder mit Cursor und Leertaste das Umschalten keineswegs rückgängig macht). Wenn man das weiß, kann man sehr virtuos damit umgehen; wenn man es nicht weiß, fühlt man sich leicht vom Gerät genarrt. Mit der Return-Taste kommt man auf jeden Fall in den direkten Modus zurück, auch mit SHIFT-RETURN: dann vermeiden Sie die Übernahme bzw. Ausführung der betroffenen Zeile.

Vom Löschen war soeben die Rede: DEL (delete) löscht sozusagen im Rückwärtsgang; Sie können damit frisch geschriebene Fehler korrigieren. Sie können aber auch mitten in einen vorhandenen Text mit dem Cursor gehen und dort mit DEL (von rechts nach links) etwas entfernen, ohne daß eine Lücke bleibt.
Auch mit der Leertaste wird etwas gelöscht, im Gegensatz zur

Schreibmaschine; zum Springen ohne Löschen gibt es ja die beiden Cursortasten (Merkregel: mit SHIFT in den umgekehrten Richtungen wie beim Schreiben, nach links bzw. oben); auch die Cursorbewegungen und die Funktionen HOME und CLR HOME werden nach einer ungeraden Zahl von Gänsefüßchen nicht ausgeführt, sondern symbolisch vermerkt.

Etwas ganz Feines, was es bei normalen Schreibmaschinen nicht gibt, ist INST (insert): links neben dem blinkenden Cursor wird ein leerer Platz eingefügt, der sich automatisch im indirekten "Gänsefüßchenmodus" befindet. Die Begrenzung der Länge von Programm-Zeilen gilt aber trotzdem.

Noch eine ganz andere Bemerkung zur Leertaste: Zwischen BASIC-Schlüsselwörtern, Zahlen und Variablennamen usw. muß man sie fast nie setzen, man spart auf diese Weise Speicherplatz; andererseits muß man aber beim Lesen wie der Computer auch unter diesen Umständen alle Schlüsselwörter wiedererkennen (z.B. BANDE bedeutet: B AND E). Es ist also eine Entscheidung zwischen dem Knautschen von Speicherplatz und Lesbarkeit (für den Menschen). In einigen Fällen muß auch der Computer durch Leerzeichen vor Mißverständnissen bewahrt werden: Wenn Sie F OR G als FORG schreiben, liest der C 64 : FOR G. - Umgekehrt können auch zuviel Leerzeichen schaden: IN PUT statt INPUT wird nicht verstanden, die einzige Ausnahme ist GOTO, wofür man auch GO TO schreiben darf.

Vor einigen Mißverständnissen, die sich aus der nicht besonders glücklichen Beschriftung einiger Tasten ergeben, muß gewarnt werden:

Die RETURN-Taste hat nichts mit der BASIC-Anweisung RETURN zu tun, ihr Name bedeutet eigentlich Rückkehr zum Zeilenanfang. Das ist aber das allerwenigste von dem, was sie tut.

Sie veranlaßt zugleich einen Sprung in die nächste Zeile. Beides zusammen heißt bei manchen Fernschreibern "Carriage return & line feed". Soll nur dieses geschehen, so ist die RETURN-Taste mit SHIFT zu drücken (also SHIFT schon etwas eher und dann gedrückt lassen). Die RETURN-Taste für sich alleine macht noch viel mehr: Befindet sich der Cursor (also das blinkende Rechteck) in einer Zeile mit einer Anweisung ohne Zeilennummer (wobei evtl. zwei Bildschirmzeilen zu einer Programmzeile verknüpft sein können), so wird die Ausführung dieser Anweisung (oder wenn mehrere in der Zeile sind, aller dieser Anweisungen) veranlaßt. Kann der C 64 in dem Text keinen Sinn erkennen, meldet er "SYNTAX ERROR". Noch ein Trick für ganz Faule: Steht der Cursor in einer Anweisung am Beginn einer Zeile, dahinter ein Doppelpunkt, gefolgt von sinnlosem Text, so klappt es auch.

Steht am Beginn der (Programm-)Zeile, in welcher der Cursor blinkt, eine (zulässige) Zeilennummer, so wird die ganze Programmzeile (vor und evtl. auch hinter dem Cursor) unter dieser Nummer ins Programm aufgenommen, ohne Rücksicht auf Fehler (die werden erst bei der Ausführung aufgespürt, falls überhaupt). Eine evtl. schon vorhandene Zeile der gleichen Nummer verschwindet zugleich. - Im laufenden Programm spielt die RETURN-Taste eine wichtige Rolle bei der Anweisung INPUT: Eine Eingabe aus mehreren Zeichen wird damit abgeschlossen und gewissermaßen abgeschickt - vorher kann man sie noch mit DEL korrigieren. Diese ganzen Funktionen der RETURN-Taste entsprechen den Tasten ENTER oder EXECUTE bei anderen Rechnern; die Aufschrift RETURN ist also eine gelinde Untertreibung.

Mißverständnisse gibt es auch bei der RUN/STOP-Taste: Sie bewirkt gemäß ihrer unteren Aufschrift normalerweise STOP, mit SHIFT zusammen jedoch nicht RUN, sondern LOAD & RUN, also Laden eines Programms von der Tonbandcassette und anschließenden Start. Man kann sich aber mit SIMON's BASIC eine der

ganz rechts angebrachten "Funktionstasten" zur RUN-Taste machen: mit der Anweisung (direkt, also außerhalb eines Programms) KEY 7,"♥RUN"+CHR$(13) (Return-Taste).

Auch die RESTORE-Taste hat nichts mit der gleichnamigen Anweisung zu tun. Sie wirkt (als Sicherung gegen versehentliches Drücken) nur zusammen mit der STOP-Taste und ist der einzige Ausstieg aus einer INPUT-Abfrage. Sie setzt den C 64 aber keineswegs in jeder Beziehung in den Zustand, als wäre er frisch eingeschaltet.

Einige Tasten (Leertaste, Cursor-Tasten, INST und DEL) haben normalerweise Dauerfunktion, d.h. bei längerem Drücken wirken sie so, als würden sie nach kurzer Wartezeit immer wieder (und dann etwas schneller) gedrückt. Beachten Sie das evtl. bei GET-Abfragen: Wenn Mehrfacheingaben stören, sollte man also nicht gerade auf das Drücken der Leertaste warten lassen. Man kann die Dauerfunktion auch abschalten oder auf alle Tasten ausdehnen:

POKE 65Ø,Ø Normalzustand
POKE 65Ø,128 alle Tasten mit Dauerfunktion
POKE 65Ø,64 keine Tasten mit Dauerfunktion

Eingaben über die Tastatur kann man auf drei Arten abfragen: mit INPUT, mit GET und mit speziellen PEEK-Abfragen, in SIMON's BASIC auch mit INKEY oder FETCH.

WAS BEDEUTEN DIE ZEICHEN ?

Festgelegte Tastaturzeichen

Sie können vom Programm her allen beliebigen Zeichen feste Funktionen zuweisen (z.B. den Klammeraffen @ als Taste für die nächste Spielrunde); einige Dinge liegen aber in BASIC fest, wenn sie außerhalb der Gänsefüßchen vorkommen:

Das Semikolon ; wirkt hinter PRINT als Trennzeichen ohne Zwischenraum, nach INPUT trennt es den Fragetext von der betroffenen Variablen.

Der Punkt . dient als Dezimalzeichen, d.h. er steht zwischen den Einern und den Zehnteln.

Das Komma , trennt gleichberechtigte Dinge, auf die eine Anweisung wirken soll, z.B. INPUT A,B,C oder DIM A(5Ø),B(36),C(24). Hinter PRINT bewirkt es als Trennzeichen zwischen Variablen zugleich Weiterspringen in die nächste Spalte, was im Falle des C 64 gleich über ein Viertel der Bildschirmbreite geht

Der Doppelpunkt : trennt Anweisungen innerhalb der gleichen BASIC-Zeile; die hinteren Anweisungen können in diesen Fällen nicht mit GOTO oder GOSUB angesprungen werden, wohl aber kann man aus einem Unterprogramm nach RETURN wieder dorthin zurück.

Die Gänsefüßchen " rahmen Strings ein, ohne selbst zu ihnen zu gehören. Zwischen ihnen werden bestimmte Tastenfunktionen nicht ausgeführt, sondern markiert; man kann das mit DEL oder der RETURN-Taste umgehen.

Das Dollar-Zeichen steht als Kennzeichen am Schluß aller Stringvariablen und Stringfunktionen.

Das Fragezeichen dient als Abkürzung für PRINT (aber nicht in PRINT#) während der Eingabe.

Das Doppelkreuz # steht am Schluß der an externe Geräte adressierten Anweisungen GET#, INPUT#, PRINT#.

Die Zeichen + - * / ↑ ( ) haben die bekannten arithmetischen Bedeutungen, ebenso die Zahl π.

Der Klammeraffe @ und das Sternchen * haben beim Betreiben der Diskettenstation spezielle Bedeutungen (vgl. Betriebsanleitungen).

UND DIE TASTEN GANZ RECHTS ?

Die Benutzung der Funktionstasten

Wenn man den C 64 ohne BASIC-Erweiterung benutzt, machen die vier Tasten ganz rechts auf der Tastatur einen ziemlich überflüssigen Eindruck, denn sie haben keine festgelegten Funktionen. Vielmehr muß man sie erst vom Programm her mit Bedeutungen belegen. Mit GET (siehe S. 36) kann man es auf zwei Arten machen:

```
10 GET A$:IF A$=CHR$(133) THEN PRINT "F1"
20 GOTO 10
```

oder:

```
10 GET A$:IF A$="■" THEN PRINT"F1"
20 GOTO 10
```

Das (reverse) ■ ist dabei das Zeichen, das auf dem Bildschirm erscheint, wenn Sie die Taste f1 hinter einem Gänsefüßchen drücken. Eine andere nützliche Abfrage verwendet PEEK(2Ø3) evtl. zusammen mit PEEK(653) (vgl. S. 37.
Sinnvoll wird das Abfragen etwa, wenn man in Abhängigkeit vom Drücken einer dieser Spezialtasten in ein bestimmtes Unterprogramm springt oder sonstwie in ein laufendes Programm eingreift (Gasgeben oder Lenken bei einem Autorennen z.B.). Es sind vier Spezialtasten, die mit SHIFT jeweils eine Zweitbelegung bekommen (mit der Commodore-Taste C= die gleiche), das sind also acht Quasi-Buchstaben. Im ASCII-Code haben sie acht verschiedene Nummern, beim Abfragen mit PEEK werden dagegen die vier Tasten und die Umschaltung mit SHIFT unabhängig voneinander erfaßt:

oben PEEK(2Ø3)= 4	f1 CHR$(133)	f2 CHR$(137)
⇕ 5	f3 134	f4 138
6	f5 135	f6 139
unten 3	f7 136	f8 14Ø
	PEEK(653)=Ø ohne SHIFT	PEEK(653)=1 mit SHIFT

Mit SIMON's BASIC kann man die Funktionstasten sehr effektiv nutzen: mit der Belegung durch KEY (DISPLAY zeigt Ihnen die vorhandenen Belegungen). Nützliche Beispiele sind:

```
KEY 1,"OPEN 1,4:CMD 1"+CHR$(13)
KEY 3,"PRINT=1"+CHR$(13)
KEY 5,"COPY"+CHR$(13)
KEY 7,"♥RUN"+CHR$(13)
KEY 8,"♥LIST"+CHR$(13)
```

Hierin ersetzt CHR$(13) das Drücken der Return-Taste, und in den beiden letzten Beispielen ist das Löschen des Bildschirms gleich eingebaut. Auch Wörter (Namen, BASIC-Schlüsselwörter etc.) können den Tasten zugeordnet werden (dann ohne CHR$(13), Gänsefüßchen müssen durch CHR$(34) ersetzt werden. Mit SHIFT und/oder C= sind so 16 Belegungen möglich.

Eine weitere Möglichkeit, die Funktionstasten mit SIMON's BASIC abzufragen, ist die Funktion INKEY:

INKEY

INKEY ist eine Funktion, die das aktuelle Drücken der Funktionstasten in Verbindung mit Umschalttasten angibt, d.h. eine der vier Spezialtasten ganz rechts muß gedrückt sein, wenn INKEY abgefragt wird. Die Ergebnisse sind für die 4 Tasten: 1 3 5 7. Ist gleichzeitig SHIFT gedrückt, ist es 1 mehr, bei der Commodore-Taste 8 mehr, und bei beiden zugleich 9 mehr. So kann man 16 Zahlen (von 1 bis 16) erhalten. Ist keine Funktionstaste gedrückt (oder gleichzeitig eine "normale" Taste), so hat INKEY den Wert Null. Mit der CTRL-Taste zusammen kann man noch mehr Zahlen erfassen, aber nicht ganz so systematisch:

Tabelle für INKEY

keine Taste: Ø

CTRL	-	-	-	-	+	+	+	+
C=	-	-	+	+	-	-	+	+
SHIFT	-	+	-	+	-	+	-	+
f1	1	2	9	1Ø	166	2Ø5	241	4
f3	3	4	11	12	168	2Ø7	243	6
f5	5	6	13	14	17Ø	2Ø9	245	8
f7	7	8	15	16	172	211	247	1Ø

VOM LOB DER FAULHEIT

Zeilennummern und Editiertricks

Zeilennummern sind ganze Zahlen von Ø bis 63999, sie dienen vor allem als Ziele für Programmsprünge: bei GOTO und GOSUB. Beachten Sie aber auch, daß nach einer nichterfüllten IF-Bedingung bei THEN an den Anfang der nächsten Zeile gesprungen wird (auch wenn in der alten Zeile mehrere Anweisungen auf das THEN folgen).

Sinnvollerweise beginnt man einzelne Programmblocks mit Vielfachen von 1ØØ oder 1ØØØ und läßt viele Nummern frei (üblicherweise benutzt man erst nur volle Zehner), damit man später noch Einfügungen machen kann. Niedrige Nummern zu verwenden, bringt keine Vorteile (außer beim Tippen): Der Speicherplatz ist für jede Zeilennummer genau zwei Bytes lang (da sie in Gestalt von zwei Bytes gespeichert wird) lediglich in den GOTO- oder GOSUB-Anweisungen, sparen Sie das eine oder andere Byte.
Man kann sich viel Arbeit (und Tippfehler) ersparen, wenn man sich klarmacht, daß die RETURN-Taste alles, was auf dem Bildschirm steht und wie eine Programmzeile aussieht (d.h. mit einer Zeilennummer beginnt), als Programmzeile einliest, wenn der Cursor sich in ihr befindet. Das bedeutet auch, daß der Cursor nicht erst an das Ende wandern muß. Sie können auch eine Zeile, die aufgrund einer Auslistung auf dem Bildschirm steht, beliebig überschreiben und in der geänderten Form mit der RETURN-Taste eingeben. Ist dabei die Zeilennummer gleichgeblieben, so ist die alte Form dieser Zeile durch die neue ersetzt worden.
Sie können aber auch die Zeilennummer (oder nur diese) ändern: Dann bleibt die alte Zeile erhalten; die mit der neuen Nummer kommt neu hinzu oder wird geändert. Diese Tricks (die z.T. nur bei Commodore möglich sind) sind insbesondere dann bequem, wenn im Programm viele fast gleichlautende Zeilen vorkommen oder wenn Sie einige Zeilen verlagern wollen.

ANHALTEN ZUM ESSENFASSEN

Eingaben mit INPUT

Format: INPUT a
INPUT t;a

mit Variable a oder Variablenliste a (worin Kommata zwischen den einzelnen Variablen stehen)
Fragetext t (dieser darf maximal 39 Zeichen lang sein)

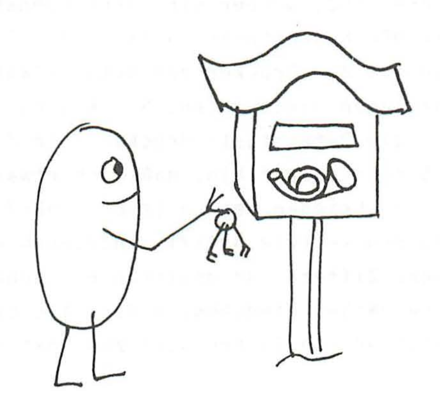
Beispiele:
```
INPUT "ZAHL DER SPIELER ";Z
INPUT "REIBUNG J/N ";R$
INPUT "ANF.-KOORD.X,Y ";X,Y
INPUT "ANF.-KOORD.X,Y ";X(J),Y(J)
```

Die Anweisung INPUT enthält also wahlweise einen Fragetext, der durch ein Semikolon von der folgenden Variablen(-liste) getrennt wird. Die Ausführung geht so vor sich: Das laufende Programm unterbricht sich und schreibt den Fragetext (falls vorhanden) auf den Bildschirm, auf jeden Fall aber ein Fragezeichen. Dahinter blinkt dann der Cursor: Der Computer verlangt nach einer Antwort und wartet darauf. Der Benutzer muß nun eine Zahl oder einen Text oder deren mehrere, und zwar passend zu der Variablenliste der Anweisung, eintippen; wenn es mehrere sind, werden sie durch Kommata getrennt. Noch sind mit DEL Korrekturen möglich. Die Eingabe wird abgeschlossen mit dem Drücken der Return-Taste: Nun wird der Brief sozusagen eingeworfen. Sie können auch nach jeder Einzeleingabe die Returntaste drücken; ein doppeltes Fragezeichen weist dann darauf hin, daß noch etwas fehlt. Zahlen können mit E geschrieben werden (z.B. 3.5E-7), es können aber keine Ausdrücke (wie 3*4+7) eingegeben werden. Texte aus Buchstaben, Ziffern und gewissen einfachen Zeichen kann man ohne Gänsefüßchen eingeben, anders ist es z.B. bei vor- oder nachlaufenden Leertasten oder zum Text gehörenden Kommata.

Wenn Sie trotzdem gerne ausführliche Fragen stellen wollen, so empfiehlt sich eine PRINT-Anweisung für den Anfangsteil der Frage. Übrigens macht es sich gut, wenn ein Fragetext auf dem Bildschirm mit Leerzeilen abgesetzt ist. Das geht am einfachsten mit einem programmierten Cursor am Anfang des Textes in der INPUT-Anweisung.

Beachten Sie, daß im Grafikmodus (HIRES z.B.) INPUT nur "im Blindflug" möglich ist, es ist also sinnvoll, Eingaben mit dieser Anweisung vor dem Einschalten der Hochauflösung vorzunehmen.

INPUT ist ein relativ umfangreicher und komplexer Befehl; er gestattet die Eingabe von Inhalten aus mehreren Zeichen, schreibt vorher den Fragetext und weist die Antworten den Variablen als (neue) Werte zu. Nachteilig ist mitunter, daß er mit einer Programmunterbrechung einhergeht. Man braucht daher auch noch andere Arten, Eingaben über die Tastatur zu machen, die ohne Unterbrechung möglich sind.



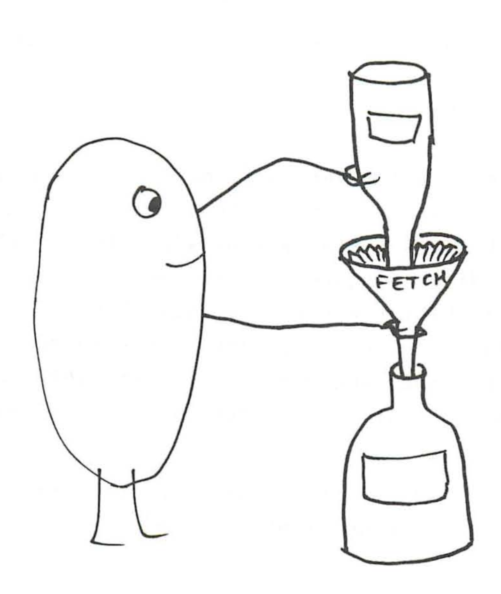
FETCH

FETCH "e",z,t$ (nur in SIMON's BASIC)

Diese Eingabeanweisung ist eine sehr nützliche Weiterentwicklung von INPUT: es werden nur die in "e" zur Auswahl vorgeschriebenen Zeichen angenommen, und zwar entweder bis die Return-Taste gedrückt wird, oder bis die Zahl z von Einzelzeichen erreicht ist. Das Ergebnis wird in die Stringvariable (hier mit t$ symbolisiert) übernommen. Ebenso wie INPUT ist FETCH eine Anweisung, die das Programm unterbricht, bis eine Eingabe stattgefunden hat, andererseits schreibt sie aber keinen Text auf den Bildschirm.Gibt man für e HOME oder ⇔ oder ⇕ , so begrenzt man die anzunehmenden Zeichen wie folgt:

HOME	Buchstaben ohne SHIFT
⇔	Buchstaben mit oder ohne SHIFT
⇕	Ziffern, grammatische und math. Zeichen

Alle übrigen Zeichen werden bei der Eingabe ignoriert.
FETCH ist also eine Eingabeanweisung mit einem "Filter".



GANZ IM VORÜBERGEHEN
Eingaben mit GET

Format: GET a
mit Variable a (String meist günstiger als Zahl)

Wird eine Taste (evtl. zusammen mit Shift- oder Commodore-Taste C=) gedrückt, so wird die zugehörige Kennzahl (ein Byte, also eine Zahl von 0 bis 255) in einen Zwischenspeicher (Tastaturpuffer) gesetzt, falls dieser nicht schon voll ist - er faßt 10 Zahlen. Wird die Anweisung GET ausgeführt, so greift sich der Computer das älteste Zeichen aus dem Puffer und weist es der Variablen a zu. Ist a eine Stringvariable (z.B. B$), so hat es nun den Inhalt, der sonst beim Drücken der Taste (bzw. ihrer Kombination mit SHIFT oder C=) geschrieben würde. Ist a aber eine Zahlenvariable, so erhält sie bei 1 bis 9 diese Werte, sonst aber stets die Null, auch wenn überhaupt keine Taste gedrückt worden war. Das Programm wird dabei keineswegs unterbrochen, vielmehr schaut GET sozusagen im Vorbeigehen im Briefkasten nach.

Soll auf eine Eingabe gewartet werden, so muß man das etwa so formulieren:

```
10 GET A$:IF A$="" THEN 10
20 A=VAL(A$)
```

"" ist dabei ein String der Länge Null, im Gegensatz zur Leertaste " ". Jetzt wartet das Programm gewissermaßen am Briefkasten, bis etwas eingeworfen wird. Anders als bei INPUT besteht diese Eingabe aber nur aus einem einzigen Zeichen oder im Rahmen einer FOR...NEXT-Schleife aus einer gegebenen Anzahl von Zeichen. Natürlich kann man sich mit GET und anderen Anweisungen eine komplette INPUT-Prozedur zusammenbauen; das ist insbesondere bei der Feingrafik interessant

Oft ist es nötig, den Tastaturpuffer rechtzeitig zu leeren:

```
30 FOR I=1 TO 10:GETL$:NEXT
```

Oder wir erzählen dem Computer einfach, sein Tastaturspeicher wäre leer (obwohl das eigentlich gar nicht stimmt):

```
POKE 198,Ø
```

GET hat den Vorteil, daß das Programm zu dieser Eingabe nicht unterbrochen werden muß; nachteilig ist gelegentlich, daß ohne weiteres nur ein Zeichen eingegeben werden kann.

Daß jeder Tastendruck nur einmal mit GET erwischt wird, GET also auf Dauerdruck nicht mehrfach reagiert (in einer Schleife), und daß es auch noch evtl. mit Verspätung stattfindet (wegen des Tastaturpuffers), kann erwünscht sein, muß es aber nicht. In diesem Beispiel:

```
1Ø GET A:IF A=7 THEN PRINT 7
2Ø FOR I=Ø TO 5ØØØ:NEXT:GOTO 1Ø
```

wird z.B. die Zahl 7 fünfmal geschrieben, wenn man die Taste 7 fünfmal nacheinander drückt. Wegen der Pausenschleife kann das Schreiben mit einiger Verzögerung geschehen.

Falls Sie (z.B. bei einem Klavierspielprogramm) nicht wünschen, daß sehr schnell gedrückte Tastenfolgen nachgearbeitet werden, können Sie den Tastaturpuffer begrenzen, sinnvollerweise meistens auf 1 (statt der normalen 1Ø, auf jeden Fall aber nicht über 1Ø hinaus):

```
POKE 649,1
```

TASTEN ALS KLINGELKNÖPFE

Tastaturabfragen mit PEEK

Oft will man aber eine Anweisung genau so lange wiederholt ausführen lassen, wie man eine bestimmte Taste drückt. Dazu eignet sich der Speicherplatz 2Ø3. In ihm befindet sich normalerweise die Zahl 64. Während des Druckes auf eine bestimmte Taste ist der Inhalt dieses Speichers eine für die jeweilige Taste kennzeichnende Zahl (unabhängig von SHIFT usw.). Mit dem Kurzprogramm

```
1Ø PRINT PEEK(2Ø3):GOTO 1Ø
```

kann man sich ein Verzeichnis anlegen. Natürlich hat auch der Computer dieses Verzeichnis: es beginnt bei Speicherplatz

60289. Ein kleines Abfrage-Programm geht z.B. so:

```
10 A=PEEK(60289+PEEK(203)):PRINT CHR$(A);A:GOTO 10
```

Die SHIFT-LOCK-Taste ist die einzige einrastende Taste; man kann sie z.B. benutzen, um ein Programm vorübergehend zu verlangsamen, indem man in eine Schleife einbaut:

```
370 IF PEEK(653)=1 THEN FOR I=0 TO 100:NEXT
```

Man kann es auch anhalten und durch Ausrasten wieder weiterlaufen lassen:

```
460 IF PEEK(653)=1 THEN 460
```

Bei Spielen für zwei Personen mit Tasten führt die normale Tastenabfrage mit GET und mit PEEK(203) dazu, daß die Partner sich gegenseitig behindern können bzw. daß sie sogar unterschiedlich gut "zum Zuge kommen" (da beim gleichzeitigen Drücken mehrerer Tasten immer die gleichen den Vorrang haben). Hier kann man der einen Person Tasten für PEEK(203) und der anderen die für PEEK(653) zuweisen; sie können sich dann nicht gegenseitig behindern, auch wenn sie überhaupt nicht mehr loslassen.

Der Speicherplatz 653 achtet sozusagen auf die drei Umschalttasten SHIFT (ebenso SHIFT LOCK), C= und CTRL: Er enthält bei SHIFT oder SHIFT LOCK eine 1, bei C= eine 2, bei CTRL eine 4 und beim gleichzeitigen Drücken von mehreren dieser drei die entsprechende Summe (oder, was dasselbe ist, aber den Ursachen näher kommt: das logische ODER von den Zahlen). Die Codierung der Tastatur sieht nun so aus:

←	1	2	3	4	5	6	7	8	9	0	+	-	£			f 1
57	56	59	8	11	16	19	24	27	32	35	40	43	48	51	0	4

CT	Q	W	E	R	T	Y	U	I	O	P	@	*	↑	RS	f 3
4	62	9	14	17	22	25	30	33	38	41	46	49	54		5

		A	S	D	F	G	H	J	K	L	:	;	=	Ret	f 5
	1	10	13	18	21	26	29	34	37	42	45	50	53	1	6

C=	Sh	Z	X	C	V	B	N	M	,	.	/	Sh			f 7
2	1	12	23	20	31	28	39	36	47	44	55	1	7	2	3

SPACE = Leertaste	keine Taste
60	64

Die inversen Zahlen (1,2,4) gelten für PEEK(653), die anderen für PEEK(203) oder PEEK(197).

IST DA JEMAND ?

Abfrage der Feingrafikpunkte

TEST(x,y) (in SIMON's BASIC)

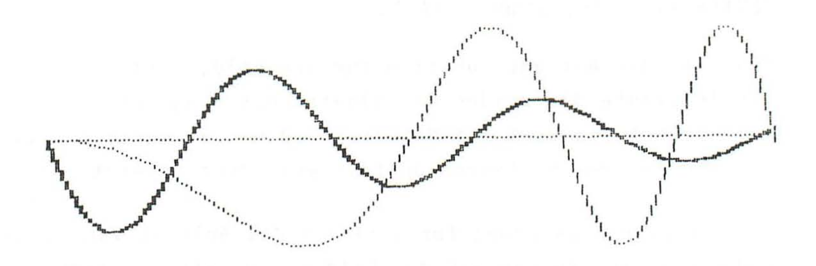
TEST ist die Abfrage-Funktion für den Bildschirm: sie hat als Argumente die beiden Koordinaten des fraglichen Punktes und liefert als Wert bei Multicolor Ø bis 3, sonst Ø oder 1, je nachdem, welche (vorgewählte) Farbe dort gesetzt ist.

Eine typische Anwendung für TEST ist das Ablesen von Schnittpunkten zweier Kurven auf dem Bildschirm, also gewissermaßen die Nachahmung einer grafischen Lösung. In Wirklichkeit ist es natürlich auch eine Rechnung, nur mit der Besonderheit, daß sie in relativ grobem Raster erfolgt und sehr primitiv vor sich geht, aber dafür zugleich anschaulich gemalt wird. Da es (leider) die Funktion TEST nur für einzelne Punkte, aber nicht für Linien gibt, muß man die zweite Kurve punktweise zeichnen (zumindest punktweise abfragen) und dabei die Punkte so dicht legen, daß kein Schnittpunkt durch die Lappen gehen kann. Dabei ist es nie ganz zu vermeiden, daß in der Nähe eines (echten) Schnittpunktes mehrere Rasterfelder von beiden Kurven "betreten" werden oder sogar dort Schnittpunkte vorgetäuscht werden, wo sich die Kurven nur sehr nahe kommen.

```
100 HIRES 1,0:MULTI 1,10,0:POKE53281,14:YA=150
105 LINE 0,150,159,150,2;:REM NULL-LINIE
107 :
108 REM 1. KURVE
109 :
110 FOR X=0 TO 159:Y=150+50*SIN(X*X/2000)
120 LINE XA,YA,X,Y,1:XA=X:YA=Y:NEXT
127 :
128 REM 2. KURVE UND ABFRAGE
129 :
130 FOR X=0 TO 159 STEP.1
140 Y=SIN(X/10)*EXP(-X/100)
150 IF TEST(X,150+50*Y)=1 THEN TEXT 0,N*10,STR$(X)+STR$(Y),4,1,8:N=N+1
160 PLOT X,150+50*Y,3:NEXT
300 GOTO 300
```

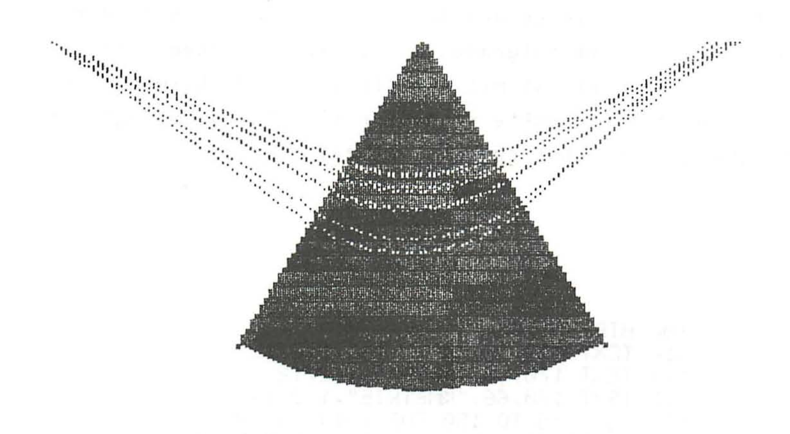
* falsch kein ;, sonst syntax Error

```
0   0
26.9000003     .3334713
74.0999978     .4304263
109.099996 -.3346513
136.09997      .2215936
```



Stimmt eine der Vordergrundfarben mit der Hintergrundfarbe überein, kann eine derart unsichtbar gemalte Struktur dennoch abgefragt werden (z.B. unsichtbare Grenze in einem Spiel).

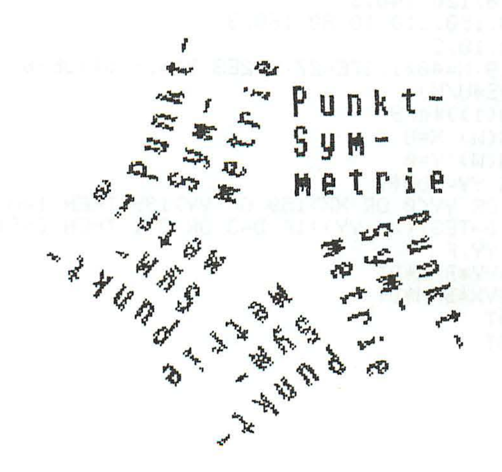
In dem folgenden Programm bewegt sich ein elektrisch geladenes Teilchen (nämlich ein Ion) durch ein hellblaues und ein hellrotes Gebiet. Diese unterscheiden sich in der magnetischen Feldstärke: das hellrote stellt den Raum zwischen den sektorförmigen Polschuhen eines starken Magneten dar, das hellblaue den übrigen Raum (im Hochvakuum). Der Ort des Ions wird weiß eingezeichnet, aber über zwei indirekte Codenummern: betritt das Ion ein bereits weißes Rasterfeld, so kann TEST trotzdem unterscheiden, ob es Vordergrundfarbe Nr. 1 oder 2 ist, und daran erkennt unser Programm, ob an der Stelle das starke Magnetfeld herrschen soll oder nicht: im Feld wird die Bahn des Teilchens gekrümmt. Wie man sieht, führt das zusammen mit der geometrischen Form des Sektorfeldes zu einer Wiedervereinigung (Fokussierung) von Ionen, die in (etwas) verschiedene Richtungen (aber gleich schnell) starten. Das Programm ist zugleich ein Beispiel für die Simulation eines physikalischen Vorganges; Sie können in Zeile 120 U oder B etwas abändern (einige 10 % z.B.) und die Folgen betrachten.



```
90 REM MASSENSPEKTROMETER
91 :
100 HIRES 1,0:MULTI 1,1,10:POKE53281,14
110 LINE 80,0,40,140,3
111 LINE 80,0,120,140,3
112 ARC 80,0,150,210,10,80,160,3
113 PAINT 80,10,3
120 E=1.6E-19:M=40*1.67E-27:U=2E3:B1=.6:DT=3E-8
130 V=SQR(2*E*U/M)
140 W=(1+RND(1))*π/9
150 VX=V*COS(W):X=0
155 VY=V*SIN(W):Y=0
160 XX=500*X:YY=900*Y
165 IF XX<0 OR YY<0 OR XX>159 OR YY>199 THEN 140
170 F=2:B=0:D=TEST(XX,YY):IF D=3 OR D=1 THEN B=B1:F=1
180 PLOT XX,YY,F
190 VX=VX+E*VY*B/M*DT
195 VY=VY-E*VX*B/M*DT
200 X=X+VX*DT
205 Y=Y+VY*DT
220 GOTO160
```

Eine spezielle Anwendung von TEST kann auch darin bestehen, Zeichen des Bildschirms, so auch mit TEXT oder CHAR erzeugte Schrift, zu duplizieren und dabei evtl. auch noch zu verformen oder, wie im folgenden Beispiel, zu drehen: die waagerechte Schrift ist mit TEXT in den Bildschirm geschrieben, die einzelnen Punkte sind dann mit TEST abgefragt und punktsymmetrisch mehrfach neu eingezeichnet.

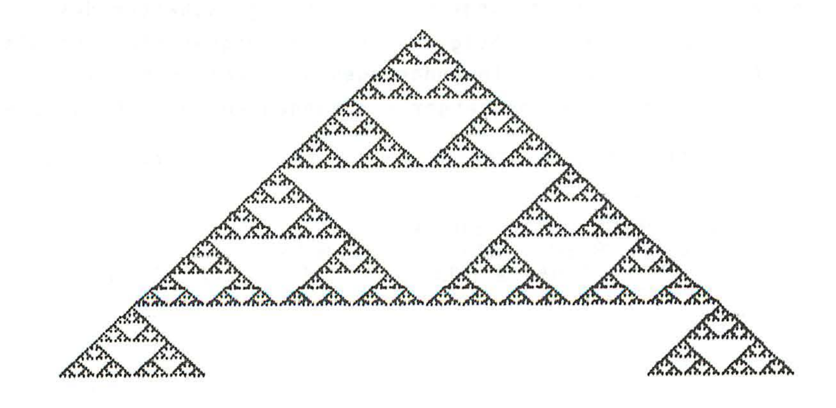
```
100 HIRES 1,0
110 TEXT 170,30,"▒▒UNKT-",1,2,12
111 TEXT 170,48,"▒▒YM-",1,2,12
112 TEXT 170,66,"▒METRIE",1,2,12
120 FOR X=5 TO 150:FOR Y=14 TO 70
121 A=TEST(160+X,99-Y):IF A=0 THEN 130
122 W=ATN(Y/X):R=SQR(X*X+Y*Y)
123 FOR I=1 TO 4
124 U=160+R*COS(W+π*I*.4)
125 V=99-R*SIN(W+π*I*.4)
127 PLOT U,V,1:NEXT
130 NEXT:NEXT
300 GOTO 300
```

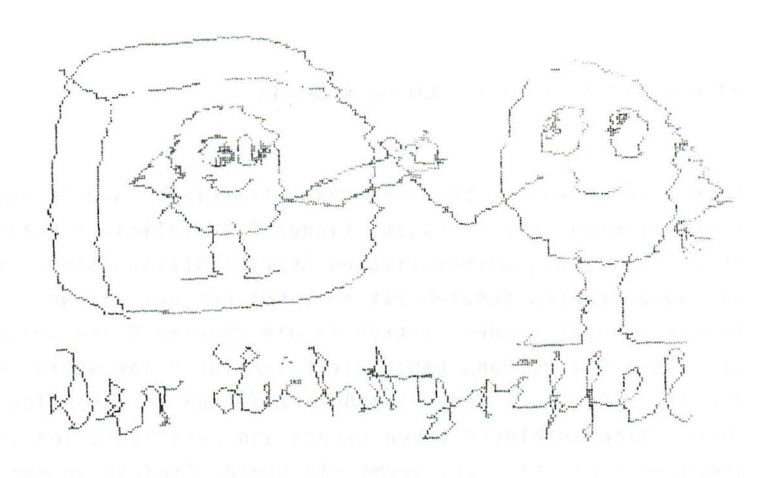


Das Pascal-Dreieck

Auch wenn Ihnen die Begriffe "Pascal-Dreieck" und "Binomial-koeffizienten" nichts sagen, finden Sie vielleicht Gefallen an dieser streng mathematischen Grafik: Stellen Sie sich ein ausgedehntes Schachbrett vor, bei dem nur die weißen Felder benutzt werden: mitten in die oberste Reihe setzen Sie einen Stein. Dann gehen Sie Reihe für Reihe weiter mit der Spielregel: in jedes (weiße) Feld, von dessen beiden oberen Nachbarfeldern (oben rechts und oben links von ihm) genau eins besetzt ist, kommt ein Stein. Einzeln in der Reihe stehende Steine sind dabei Ausgangspunkte für zwei Reihen, endständige werden dagegen nur nach außen fort-gesetzt. Insgesamt erbibt sich ein ästhetisch reizvolles Muster, das sich im Großen immer weiter wiederholt (nach unten geht es beliebig weit), und das zwar sehr streng aussieht, aber doch nicht auf den ersten Blick erkennen läßt, wie simpel die Spielregel ist, auf der es beruht !

```
90 REM UNGERADE ZAHLEN IM PASCAL-DREIECK
91 :
100 HIRES 1,0
110 PLOT 160,0,1
120 FOR Z=1 TO 159
130 FOR X=160-Z TO 160+Z STEP 2
140 F=TEST(X-1,Z-1)+TEST(X+1,Z-1)AND1
150 PLOT X,Z,F:NEXT:NEXT
160 GOTO 160
```





Wenn man dem C 64 eine bestimmte Stelle auf dem Bildschirm zeigen will, so geht das am unmittelbarsten mit dem Lichtgriffel. Das ist einfach ein lichtempfindliches Gerät in Form eines Stiftes, das an den Port 1 angeschlossen werden muß. Hält man es mit der Spitze direkt auf den Bildschirm (der dazu hell eingeschaltet sein muß und auch an der entsprechenden Stelle eine helle Farbe zeigen muß), so kann der Computer aus den Zeitpunkten, zu denen der Elektronenstrahl in der Fernsehröhre diese Stelle aufleuchten läßt, auf die Koordinaten schließen. Das geht in Y-Richtung sehr genau, in X-Richtung hängt es von den Eigenschaften des Lichtgriffels ab. Das folgende Programm eignet sich gut als Test (bis auf eventuelle Änderungen der Zahlenwerte in der Zeile 130, die vom Fernsehgerät abhängen können). Es zeichnet

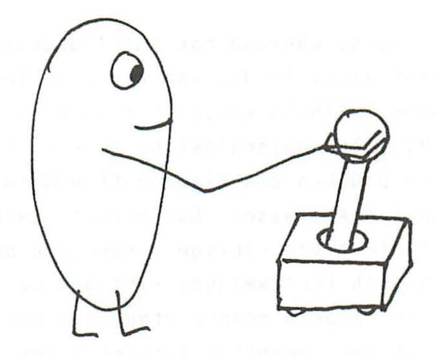
```
100 REM LICHTGRIFFEL AN PORT 1 UND SIMONS BASIC
110 :
120 HIRES 6,1
130 X=2*PENX-80:Y=PENY-45
140 LINE XA,YA,X,Y,PEEK(653):XA=X:YA=Y
150 GET H$:IFH$="@"THEN COPY:REM DRUCKER VC 1525
160 GOTO 130
```

die Bewegungen nach, während man SHIFT drückt oder SHIFT LOCK eingerastet läßt; in der übrigen Zeit löscht es. Das Programm verwendet SIMONS BASIC. Ist auch noch ein Drucker VC 1525 oder VC 1515 angeschlossen, so kann man das fertige Kunstwerk durch Drücken des Klammeraffen @ (mit COPY) auf das Papier übertragen lassen. Das direkte Zeichnen auf den Bildschirm ist nicht die einzige Verwendung des Lichtgriffels: man kann auch (Verzweigungs-)Fragen des Programms statt mit Tasten dadurch beantworten, daß man mit dem Griffel auf ein Feld mit der gewählten Antwort zeigt. Dabei fällt auch die technisch bedingte Ungenauigkeit der Koordinaten nicht so auf. Weitere Programmiervorschläge: Sie tippen mit dem Lichtgriffel auf ein Notenliniensystem, und der C 64 spielt die Musik dazu, oder: Bilder-Memory (aber mit richtigen Bildern und nicht nur Kombinationen aus Zahlen und Spielkartensymbolen !), oder: der Computer zeichnet ein (fertiges) Labyrinth und bewertet Ihre Versuche, den Lichtgriffel durch dieses hindurchzubewegen, ohne anzustoßen (Abfrage mit TEST). Schließlich kann man den Lichtgriffel auch bei geometrischen Aufgaben verwenden, um einzelne Punkte unmittelbar einzugeben, an denen dann etwas Bestimmtes geschehen soll (z.B. ein Lot errichten usw.).

DAS VERGNÜGUNGSSTEUER

auch JOYSTICK genannt

An den C 64 können zwei Joysticks gleichzeitig angeschlossen werden, das sind Richtungsschalter für jeweils 8 verschiedene Richtungen. Sie bestehen intern aus (jeweils) einem Mittelkontakt, der wahlweise an einen von vier Außenkontakten oder gleichzeitig an zwei benachbarte von diesen angeschlossen wird, indem man einen Knüppel entsprechend betätigt. Außerdem gibt es an jedem Joystick noch einen



roten Klingelknopf (Fire button). Insgesamt liefert jedes Joystick also 18 verschiedene Zustände, die im Computer als Zahlen erfaßt werden. Zum Auffinden verwenden Sie die Kurzprogramme:

1Ø PRINT 255-PEEK(56321):GOTO 1Ø [für Port 1],

bzw. 1Ø PRINT 127-PEEK(5632Ø):GOTO 1Ø [für Port 2]

bzw. 1Ø PRINT JOY:GOTO 1Ø [für Port 2 bei Verwendung von SIMON'S BASIC].

Die Ergebnisse sehen so aus:

<table>
<tr><td colspan="5">mit PEEK</td><td colspan="5">mit SIMON'S BASIC</td></tr>
<tr><td colspan="2">dezimal</td><td colspan="3">binär</td><td colspan="5">dezimal</td></tr>
<tr><td colspan="2">oben</td><td colspan="3">oben</td><td colspan="5">oben</td></tr>
<tr><td rowspan="3">links</td><td>5 1 9</td><td rowspan="3">rechts links</td><td>Ø1Ø1 ØØØ1 1ØØ1</td><td rowspan="3">rechts</td><td rowspan="3">links</td><td>8 1 2</td><td rowspan="3">rechts</td></tr>
<tr><td>4 Ø 8</td><td>Ø1ØØ ØØØØ 1ØØØ</td><td>7 Ø 3</td></tr>
<tr><td>6 2 1Ø</td><td>Ø11Ø ØØ1Ø 1Ø1Ø</td><td>6 5 4</td></tr>
<tr><td colspan="2">unten</td><td colspan="3">unten</td><td colspan="5">unten</td></tr>
</table>

In SIMONS BASIC werden die Richtungen also einfach rechtsweisend durchnumeriert, bei der direkten Abfrage gibt es dagegen eine nützliche Ordnung, die in der Binärdarstellung deutlich wird: jedes Bit bedeutet eine der vier Grundrichtungen: 1 oben, 2 unten, 4 links, 8 rechts. Aus den "diagonalen Richtungen" kann man sie mit dem logischen AND herausfischen, z.B.:

```
2Ø IF((255-PEEK(56321)AND 4)=4 THEN PRINT"LINKS"
```

Hier wird das Wort "LINKS" immer geschrieben, wenn das Joystick in Port 1 in eine der drei linken Richtungen gedrückt wird (also auch links oben oder links unten). Der rote Knopf bewirkt in den Speicherplätzen, daß das Bit mit der Bedeutung 2^4 auf Ø gesetzt wird, also:

```
30 IF (PEEK(56320)AND 16)=0 THEN PRINT"KNOPF"
```

In SIMONS BASIC dagegen wird bei gedrücktem Knopf 128 zu der vorhandenen Zahl zugefügt.

Die Abfrage des Joystick kollidiert mit der Tastatur. Wenn man also keine weiteren Vorkehrungen trifft, sollte man entweder die Tasten meiden, wenn der Joystick abgefragt wird, oder aber gezielte Anwendungen dieser Tatsache programmieren. Man hat immerhin die Möglichkeit, fast 256 Kombinationen von gleichzeitig gedrückten Tasten abzufragen: jede der folgenden Tasten liefert dabei 1 Bit, die Kombination ergibt sich aus der Summe:

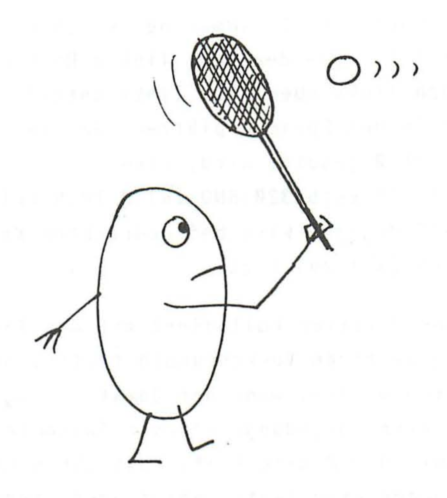
```
10 PRINT 255-PEEK(56321):GOTO 10
```

Bit Nr.	Ø	1	2	3	4	5	6	7
Beitrag	1	2	4	8	16	32	64	128
Taste	1	←	CTRL	2	Leert.	C=	Q	STOP

Drückt man also z.B. die Commodore-Taste C= zugleich mit dem Q, so gibt es 96. Die Zahl 128 kann man allerdings nicht bekommen, weil die STOP-Taste <u>allein</u> das Programm abbricht.

Bei diesem Spiel mit den Tasten braucht kein Joystick angeschlossen zu sein. Umgekehrt kann ein Joystick bei einer GET- oder INPUT-Eingabe Tastendrücke vortäuschen, was im allgemeinen unerwünscht ist.

Sind statt der Joysticks Paddles angeschlossen, so werden deren Klingelknöpfe mit den hier beschriebenen Abfragen erfaßt: der Knopf am Paddle Ø als "links", der am Paddle 1 als "rechts". Der Grund liegt einfach darin, daß es sich um die gleichen Stifte an den 9-poligen Steckern handelt. Bei SIMONS BASIC kann man jedoch nicht die Potentiometer und die Knöpfe an den gleichen Paddles abfragen, da hier die Potentiometer an Port 1 und die Knöpfe (bzw. das Joystick) am Port 2 abgelesen werden.



NICHT NUR TENNIS

Potentiometer, genannt Paddles

An den C 64 kann man ein bis zwei Paare von "Paddles" anschließen, das sind externe Drehwiderstände, die von Analog-Digital-Wandlern im Computer abgelesen werden und dabei Zahlen von 255 bis Ø (von links nach rechts) geben. Die Bezeichnung "Paddle" geht vermutlich auf die Tennisschläger bei der ersten Generation der Telespiele zurück.

Schließt man nur ein Paar an Port 1 des C 64 an, so ist die Abfrage ziemlich einfach:

```
1Ø PRINT PEEK(54297),PEEK(54298):GOTO 1Ø
```

bzw. bei SIMONS BASIC:

```
1Ø PRINT POT(Ø),POT(1):GOTO 1Ø
```

liefert nebeneinander die Ablesung der Paddles Ø und 1, dabei sollte man jedoch die Tastatur nicht gleichzeitig benutzen. Wie man (jedoch nicht bei SIMON) auch die roten Knöpfe ablesen kann, steht im vorigen Kapitel über die Joysticks. Mit geeigneten Maschinenprogrammen kann man auch zwei Paare von Paddles zugleich nutzen, mitsamt ihren roten Knöpfen (vgl. Reference Guide oder "C 64 intern").

Abgesehen von den Endbereichen ordnet der Computer den eingestellten Winkeln an den Paddle-Drehknöpfen Zahlen zu, und zwar (leider) von rechts nach links Ø bis 255 (also 2^8 verschiedene, die Auflösung ist also 8 bit). Wenn Sie lieber die großen Zahlen bei der Drehung nach rechts haben wollen (wie z.B. beim Lautstärkeknopf eines Radios),verwenden Sie z.B. 255-POT(Ø) , und wenn Sie Zahlen von Ø bis 1 wünschen, nehmen Sie (1-POT(Ø)/255). Die natürlichste Verwendung der Paddles besteht wohl darin, daß man in einem laufenden Programm, das einen Vorgang simuliert, eine Größe mit der Hand verstellt, die man auch in der Wirklichkeit an einem Drehknopf einstellt oder zumindest an einem stufenlosen Stellhebel (Gaspedal, Lenkrad, Lautstärke, Soll-Temperatur etc.).

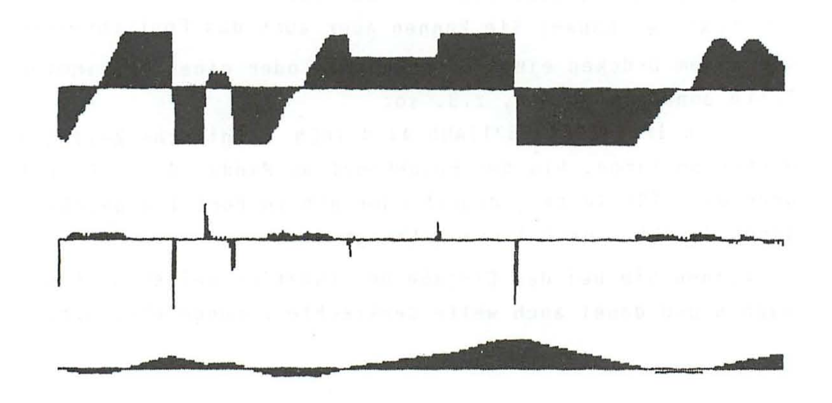
Wenn Sie ein Programm haben, das gewisse mathematische Operationen mit vorgegebenen Funktionen vornimmt (Aufsummieren, Steigung bestimmen, durch Potenzreihen oder Fourier-Reihen annähern o.ä.), so haben Sie mit den Paddles eine Möglichkeit, anstelle von formelmäßig bestimmten Funktionen auch ganz willkürliche einzugeben, ohne daß Sie dabei Hunderte von Zahlen eintippen müßten: Lassen Sie in einer FOR-NEXT-Schleife oder einer mit GOTO geschlossenen Schleife X durch die natürlichen Zahlen bis zu einem oberen Ende XM laufen und weisen Sie dabei der Funktion Y(X) einen vom Paddle bestimmten Wert zu, z.B.Y(X)=199-POT(Ø)*199/255.Damit das Ganze nicht zu schnell geht, können Sie noch eine kurze Pausenschleife FOR J=Ø TO 3ØØ:NEXT einbauen; Sie können aber auch das Fortschreiten von x vom Drücken eines Feuerknopfes oder einer bestimmten Taste abhängig machen, z.B. so:

n IF (PEEK(56321)AND 4)=4 THEN n (gleiche Zeilen-Nr. n) wartet so lange, bis der Feuerknopf an Paddle Ø (in Port 1) oder die CTRL-Taste gedrückt oder ein in Port 1 angeschlossenes Joystick nach links betätigt wird.

So können Sie bei der Eingabe der Funktion beliebige Pausen machen und dabei auch weite senkrechte Sprünge einbauen.

```
100 REM  MIT SIMON UND PADDLES AN PORT 1
101 :
102 REM  BITTE PADDLE NR.0 DREHEN UND
103 :
104 REM  ZEITWEISE FEUERKNOPF NR.0 DRUECKEN !
105 :
110 HIRES 1,0
120 DIM Y(319)
130 Y=128-POT(0)
140 PLOT X,30-Y/5,1
150 FOR J=0 TO 50:NEXT
160 PLOT X,30-Y/5,0
170 IF (PEEK(56321)AND 4)=4 THEN 230
180 IF I=0 THEN 220
190 Y(I)=Y:LINE X,30,X,30-Y(I)/5,1
200 D=Y(I)-Y(I-1):LINE X,100,X,100-D/5,1
210 S=S+Y(I):LINE X,160,X,160-S/500,1
220 X=X+1:I=I+1
230 IF I<320 THEN 130
240 GOTO 240
```

Dieses kleine Programm zeichnet eine willkürliche Kurve, während Sie den Feuerknopf Nr.Ø drücken. Die Höhe des jeweils neu hinzukommenden Punktes können Sie durch Drehen am Paddle Nr.Ø bestimmen und zugleich auf dem Bildschirm beobachten. Das Programm zeichnet auch gleichzeitig ein Steigungshistogramm (also eine Annäherung an die "Ableitung" der Differentialrechnung) und ein Summen-Histogramm (eine Annäherung an das Integral). Außerdem hält es die 160 eingegebenen Funktionswerte in einer indizierten Variablen Y(I) fest (was innerhalb des Beispiels nicht nötig wäre): Man kann also in zusätzlichen Programmteilen rechnerisch auf alle 160 Werte zugreifen (z.B. die Kurve durch Potenzreihen oder Fourier-Reihen darstellen).

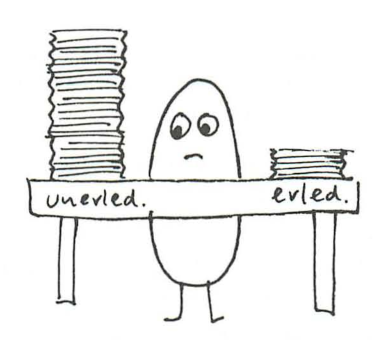


STAPELWEISE DATEN

READ und DATA

Format: DATA d oder DATA d1,d2,d3 etc.
READ v oder READ v1,v2,v3 etc. (meist jedoch als indizierte Variable in FOR-NEXT-Schleife)

Dieses Anweisungspaar gehört zu den entbehrlichen, aber oft bequemen. Wenn vielen Variablen schon vom Programm her bestimmte Werte (Zahlen oder Strings, d.h. Texte) zugewiesen werden sollen, so ist das mit der Wertzuweisung (wie: A(1)=7:A(2)=13 usw.) möglich, aber umständlich. Hier kann man READ und DATA einsetzen. Die Daten werden einfach der Reihe nach in DATA-Zeilen geschrieben,jeweils durch Kommata getrennt (aber nicht vor dem ersten und nicht nach dem letzten Datum einer DATA-Anweisung !). Stringvariable mit



vor- oder nachlaufenden Leertasten oder mit Sonderzeichen müssen in Gänsefüßchen " " eingebettet werden (die dann nicht zum jeweiligen String gehören !). Soll in einem String ein Komma, Doppelpunkt oder Semikolon (,:;) vorkommen, muß das ganze String in Gänsefüßchen " " gesetzt werden.

Mit der READ-Anweisung werden nun die Daten der Reihe nach den Variablen zugewiesen. Auf die Reihenfolge hat der Programmierer aufzupassen.(Das ist besonders dann etwas kitzlig, wenn READ in Unterprogrammen vorkommt: Die DATA werden in der Reihenfolge genommen, in der sie im gesamten Programm stehen, die READ-Anweisungen werden natürlich in der Reihenfolge behandelt, die durch GOTO oder GOSUB beeinflußt wird; es hilft also nichts, jede DATA-Serie gleich vor oder hinter die zugehörige READ-Serie zu schreiben !) Auch muß der Programmierer dafür sorgen, daß die DATA nicht früher aufgebraucht sind als die READ-Anweisungen (Fehlermeldung: OUT OF DATA), und wenn hinter READ eine Zahlenvariable steht, muß auch das Datum, das an der Reihe ist, eine Zahl sein (z.B. 3.7E-24).

Zur Verwendung der Anweisungen soll hier ein kurzes Beispiel genügen, obwohl der Sinn der Sache erst bei längeren Datenlisten zum Vorschein kommt

```
10 DATA ANTON,23,BERTA,18,CAESAR,32
20 FOR I=1 TO 3:READ N$(I),A(I):NEXT
```

N$(2) ist von jetzt an "BERTA", A(3) ist 32.

Um aus einer langen Kette von Daten eins per Zufall auszuwählen, kann man so vorgehen:

```
10 DATA ......
20 Z=100*RND(0):FOR I=1 TO Z:READ A:NEXT
```

Jetzt hat A einen der ersten 100 Werte, die hinter DATA eingegeben sind, bei jedem Lauf des Programms neu vom Zufall bestimmt. Dieses Verfahren braucht weniger Speicherplatz als eine Zuweisung aller 100 Werte an eine indizierte Variable, von der der Zufallsgenerator dann einen Index herauspickt.

UND NUN WIEDER VON VORNE !

RESTORE

Format: RESTORE

Sie können in einem Programm ganz verschiedene Daten per DATA zuweisen. Es kommt dabei nicht darauf an, ob die zugehörigen READ-Anweisungen gleich dahinter oder davor oder ganz woanders stehen; nur die Reihenfolge der Bearbeitung der READ-Anweisungen muß mit der Reihenfolge der DATA-Anweisungen zusammenpassen. Der Rechner merkt sich mit einem "Zeiger", wie weit er die DATA aufgebraucht hat. Soll er ganz von vorne anfangen, so können Sie RESTORE eingeben; der Zeiger ist dann wieder auf Null. Wollen Sie auf ein Datenpaket mehrmals in einem Programm zugreifen (z.B. zusammen mit dem Zufallsgenerator), so stellen Sie diese Daten zweckmäßigerweise an den Anfang der DATA-Sammlungen und verwenden Sie vorher jeweils RESTORE.

RESET

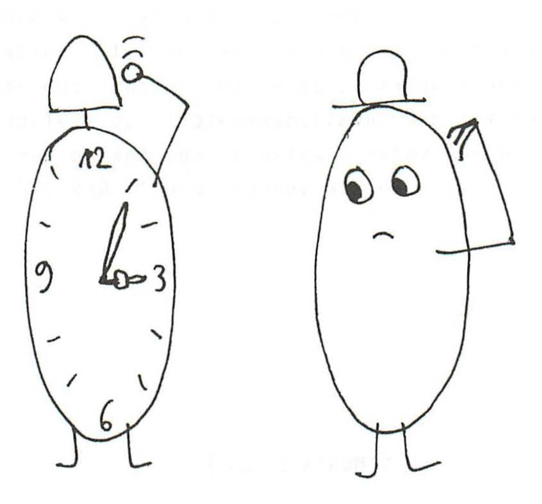
RESET n (nur in SIMON's BASIC)

Die SIMON-Anweisung RESET n bewirkt ein RESTORE nicht an den Anfang der Datensammlung, sondern an den Anfang der Zeile Nr. n. Auf diese Weise kann man auch in einem Programm mit mehreren inhaltlich verschiedenen DATA-Teilen mehrfach oder in willkürlicher Reihenfolge mit READ auf die jeweils passenden DATA zugreifen: Man setzt einfach RESET, bevor die Schleife mit READ (oder eine einzelne READ-Anweisung) folgt.

IM LAUFE DER ZEIT

Die Zeitvariable TI

Was hat der C 64 mit einer Armbanduhr gemeinsam ? Beide können die Zeit messen. Vom Einschalten des Gerätes an zählt eine spezielle Variable TI in jeder Sekunde um 60 Einheiten weiter, jedenfalls wenn nicht durch die Benutzung der Datasette (des Cassettenrecorders also) Unordnung entsteht. Man kann also durch Abfragen von TI die Zeit auf 1/60 Sekunden genau erfahren, auch im Programm zwischen zwei Anweisungen. Geben Sie z.B. ein: 1Ø PRINT TI:GOTO 1Ø



Wenn man wissen will, wie lange eine Programmschleife dauert, kann man es etwa so machen:

```
1Ø T2=TI:DT=(T2-T1)/6Ø:T1=T2
.......
9Ø GOTO 1Ø
```

In der Schleife hat DT dann den Zahlenwert der letzten Schleifendauer in Sekunden und steht für Rechnungen zur Vefügung (Achtung: Im ersten Schleifendurchlauf ist DT im allgemeinen viel zu groß; um das zu vermeiden, muß vor der Schleife einmal T1=TI abgefragt werden).

Beachten Sie bitte, daß TI eine Variable ist, die sich "ganz von selbst" ändert, indem sie die Zeit anzeigt. Sie kann indirekt auf Zahlenwerte gesetzt werden, indem man TI$ auf einen entsprechenden Wert setzt (vgl. folgenden Abschnitt). Die Variablen T2 und T1 im vorigen Beispiel sind Ablesungen von TI zu bestimmten Zeitpunkten im Programmablauf.

MINUTEN UND STUNDEN

Die Zeitvariable TI$

TI durch 6Ø teilen, geht ja noch an, die Umrechnung in Minuten und Stunden ist da schon etwas umständlicher. Zum Glück gibt es auch die String-Variable TI$, die ähnlich wie TI von selbst weiterläuft, aber auch gesetzt werden kann. Sie ist immer sechsstellig; die ersten beiden Stellen geben die Stunden an, die mittleren die Minuten, die letzten die Sekunden. Um dem C 64 mitzuteilen, daß es 14 Uhr 23 min und 4Ø Sekunden ist, schreibt man (ohne Zeilennummer):

```
TI$="14234Ø"
```

und drückt genau zur aufgeschriebenen Uhrzeit auf die RETURN-Taste. Machen Sie es mit der jetzigen Uhrzeit und geben Sie dann ein: `1Ø PRINT TI$:GOTO 1Ø`

Wenn Sie mit LEFT$ und MID$ Stunden oder Minuten herausgreifen und allerhand schönem Lärm oder den Westminster-Schlag programmieren, können Sie eine Digitaluhr mit regelmäßigem Gong oder Weckeinrichtung nachahmen, Mit der Grafikanweisung LINE können Sie auch eine richtige Analoguhr mit Stunden-, Minuten- und Sekundenzeiger auf dem Bildschirm laufen lassen. Eigene Telespiele können Sie über die Spieldauer begrenzen:

```
1Ø INPUT"SPIELDAUER IN MIN ";D:TI$="ØØØØØØ"
....
8Ø IF VAL(MID$(TI$,3,2))>=D THEN PRINT"GAME OVER":END
```

Die Genauigkeit der Zeitabfrage mit TI oder TI$ ist allerdings nicht überwältigend gut (bis 1/2 Stunde Abweichung pro Tag).

SPITZ PASS AUF

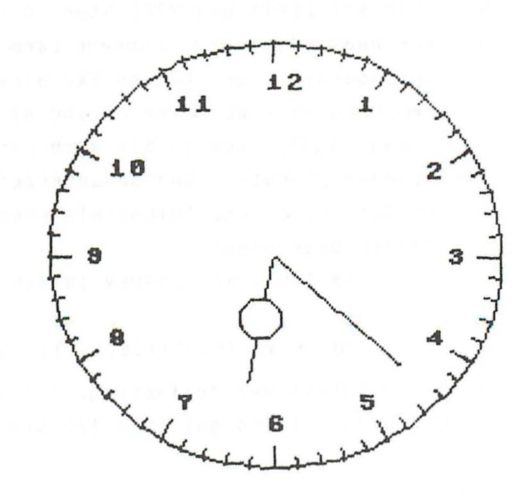
Messung der Reaktionszeit

Eine hübsche Anwendung von TI ist die Messung der Reaktionszeit:

```
90 REM REAKTIONSZEIT
91 :
100 POKE 649,1
110 FOR I=0 TO 1000+1000*RND(1):NEXT
120 PRINT"●":T1=TI:GET A$
130 GET A$:IF A$="" THEN 130
140 T2=TI:PRINT INT((T2-T1)*1000/60):GOTO110
```

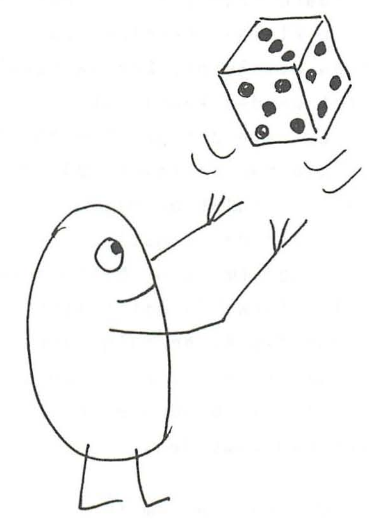
Zeile 100 begrenzt den Tastaturpuffer auf eine Eingabe, die dann sicherheitshalber in Zeile 120 auch noch leer gemacht wird, damit niemand mogeln kann und zu frühes Drücken belohnt bekommt. 110 ist eine Pausenschleife von ungewisser Länge. Nach dem Schreiben des vollen Kreises wartet der Rechner in Zeile 130 auf eine Reaktion der Versuchsperson. Leider ist das Ergebnis nur auf 1/60 Sekunden genau, was bei Werten um 200 Millisekunden stört.

Die passende Feingrafik für eine Uhr gibt es z.B. in SIMON's BASIC:



WIE DER ZUFALL SO SPIELT

RND()



Format: RND(Ø) oder RND(1)

Es gibt hauptsächlich zwei Fälle, in denen man zufällige Zahlen haben möchte: Zum einen, wenn zufällige Ereignisse (Glückspiele, Kernphysik, Genetik usw.) nachgeahmt werden sollen, zum anderen, wenn man die "Monte-Carlo-Methode" als mathematischen Trick verwendet. Statt alle Möglichkeiten durchzuprobieren, läßt man den Zufall entscheiden und wertet das Ergebnis statistisch aus.

Nun ist alles im Computer genau geregelt, sogar die Rechenfehler. Während ein Mensch auf einem Rechenschieber bei der gleichen Aufgabe immer etwas andere Abweichungen in der Ablesung macht, reproduziert der Rechner auch seine Fehler immer gleich. Es ist darum gar nicht einfach, wirklich zufällige Zahlen zu erzeugen. Der C 64 bietet uns zwei Verfahren an, die unterschiedliche Nachteile haben und die man notfalls kombinieren muß.

Die Funktion RND(1) liefert jedesmal eine Zahl zwischen Ø und 1, ohne erkennbare Regel, also gewissermaßen zufällig. Es kommen auch alle kleinen Bereiche zwischen Ø und 1 recht genau gleich oft an die Reihe. Der Nachteil (der aber auch als Vorteil genutzt werden kann) ist: Nach dem Einschalten des Gerätes ist die erste dieser (Pseudo-)Zufallszahlen stets .185564Ø16, die zweite immer .Ø468986348 usw. Wenn man eine Monte-Carlo-Simulation mitsamt ihren Zufallszahlen wiederholen möchte, ist das günstig, ebenso, wenn man bei einem Glückspiel-Programm seine Mitmenschen betuppen will: Nach einigen Spielen (jeweils mit frisch eingeschaltetem Computer) kennt man die Reihenfolge der "gewürfelten" Zahlen. Andere Argumente als 1 zwischen den Klammern von RND() ändern auch nichts, wenn sie positiv sind, wohl aber die Ø, der wir uns nun zuwenden:

RND(Ø) wird zwar auch nach einem festen Schema gerechnet, aber es startet mit der internen Uhrzeit, also mit der Zeit zwischen Einschalten des Rechners und Bearbeitung der RND(Ø)-Funktion, die natürlich von der Zeit zwischen Einschalten und Starten des Programms abhängt und damit wirklich vom Zufall. RND(Ø) hat einen anderen Nachteil: Die Zahlen sind nicht so gleichmäßig im Intervall zwischen Ø und 1 verteilt. Solange man nur 6 oder 37 verschiedene Zahlen würfeln will, spielt das keine entscheidende Rolle, wohl aber bei der Multiplikation mit größeren Zahlen (z.B. Telefonnummern). Man kann sich notfalls damit helfen, daß man die Dezimal-Ziffern einzeln mit RND(Ø) würfelt, wenn es auf eine genaue Gleichverteilung ankommt. Ein anderer Trick ist die Verwendung von RND(1) mit einem vorgeschalteten RND(Ø), das dann nicht weiterverwendet wird, aber als Ausgangsbasis für RND(1) dient und dieses damit "unberechenbar" macht, etwa so: 1Ø A=RND(Ø):A=INT(1+6*RND(1)):PRINT A

Diese Zeile liefert eine (ganze) Zufallszahl zwischen 1 und 6 (beide einschließlich), die man nicht vorhersagen kann.

Rechteckverteilungen

Wie man Zufallszahlen macht, die gleichmäßig in ein beliebiges Intervall zwischen den Zahlen A und B fallen, ist nicht sehr schwer: 1Ø Z=A+(B-A)*RND(1)

Sind A und B ganz und sollen auch nur ganze Zahlen getroffen werden (A und B beide einschließlich), so ist zu setzen:

```
1Ø Z=A+INT((B+1-A)*RND(1))
```

Fast beliebige Verteilungen

Soll die Wahrscheinlichkeit nicht in einem Intervall gleichmäßig (konstant) sein, sondern irgendeiner vorgegebenen Funktion F(X) entsprechen, so kann man nach John von Neumann ein sehr listiges Verfahren verwenden. F(X) darf allerdings in dem behandelten Intervall nirgends unendlich werden (auch nicht negativ); es sei FM ein Wert, der nirgends überschritten wird. Wir nehmen für das Beispiel an, daß die Funktion der Betrag der Summe von Sinus und Cosinus sei, was sicher nirgends größer als 2 wird, und daß das Intervall von A bis B reiche:

```
1Ø FM=2
2Ø Z=A+(B-A)*RND(1)
3Ø IF ABS(SIN(Z)+COS(Z))<FM*RND(1) THEN 2Ø
4Ø PRINT Z:GOTO 2Ø
```

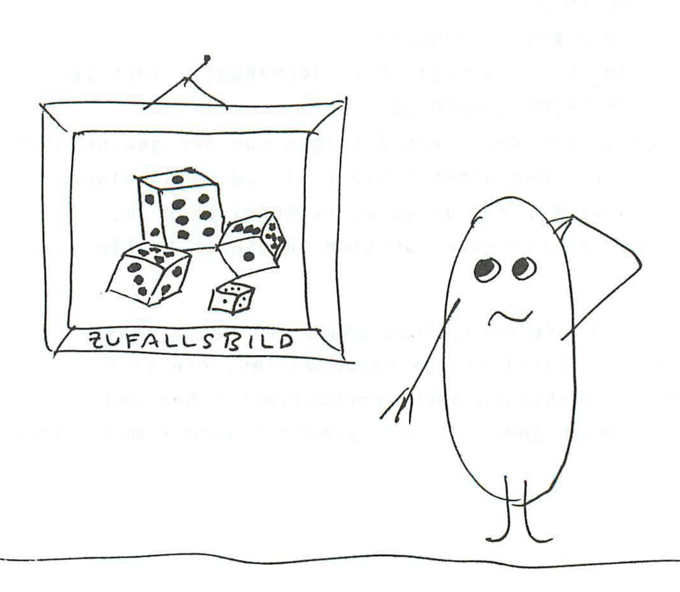
Die ausgedruckten Werte von Z folgen nun der gewünschten Verteilung. Zur Begründung: Die Zeile 3Ø weist einen gewürfelten Wert für Z um so wahrscheinlicher ab, je kleiner die angestrebte Funktion an dieser Stelle ist.

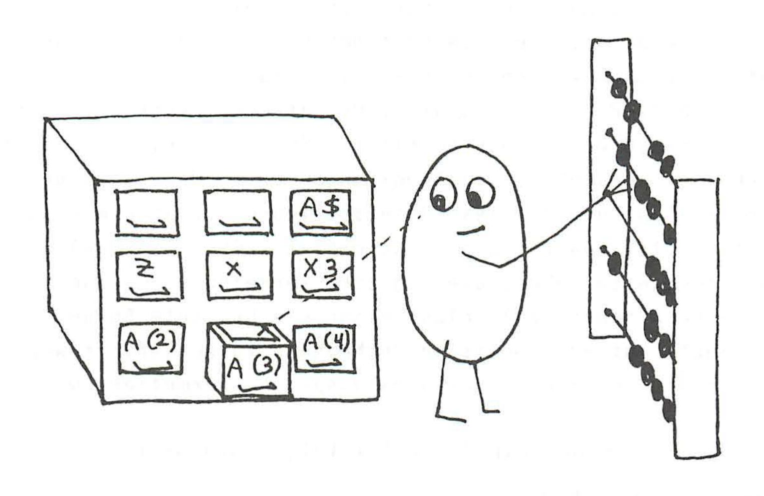
Falls Sie im Laufe eines Programmes bei jedem Lauf eine bestimmte Zufallsfolge haben wollen, die sich bei einer Wiederholung auch reproduziert (ohne daß der Rechner deswegen jedesmal ausgeschaltet werden muß), können

Sie die Ausgangsbasis für die mit RND(1) erzeugten Zufallszahlen in die Plätze 139 bis 143 hineinPOKEn. Das folgende Programm zeichnet für gleiche Eingaben (von zweistelligen Zahlen etwa) jeweils das gleiche "Zufallsbild":

```
10 FOR I=0 TO 4:INPUT A:POKE 139+I,A:NEXT
20 HIRES 1,0
30 FOR I=0 TO 10
40 LINE 320*RND(1),200*RND(1),320*RND(1),200*RND(1),1:NEXT
50 GOTO 50
```

Mit RND(Ø) gibt es jedoch auch bei gleichen Eingaben jedesmal ein anderes Bild.
RND(1) ist also eine sehr streng festgelegte Sache. Zufällig ist daran nur soviel, daß man die Reihenfolge praktisch nur dann vorhersagen kann, wenn man sie schon einmal mitgeschrieben hat (etwa wie ein Telefonbuch, worin die Nummern zwar auch festliegen, aber im Gegensatz zu einer Logarithmentafel nicht ohne weiteres berechenbar sind).





VERARBEITUNG VON DATEN

In diesem Abschnitt geht es um die Schritte, die zwischen der Eingabe und der Ausgabe im Computer geschehen:

1. Rechnungen im üblichen Sinne (Arithmetik)
2. Stringverarbeitung und Umwandlungen zwischen Strings und Zahlen
3. Vergleichsoperationen
4. Logische Verknüpfungen (Boole-Algebra)
5. Sprünge, bedingte Anweisungen, Schleifen
6. direktes Lesen und Schreiben im Speicher (PEEK und POKE)

Formal könnte man diese Dinge auch aufteilen in Funktionen. Das sind Daten, die nach bestimmten Spielregeln aus anderen Daten gewonnen werden, hier speziell aus je einem Datum, z.B. SIN(), RND() oder SGN() , Operationen (Verknüpfungen von mehreren Daten nach bestimmten Spielregeln, z.B. Addition oder logisches ODER) und Anweisungen (z.B. Wertzuweisung, Sprungbefehl usw.). Diese Unterscheidung ist aber mehr grammatisch als inhaltlich gerechtfertigt: SGN( ) wird als Funktion geschrieben, die logische Verneinung NOT aber nicht. Sie werden sehen, daß selbst Dinge wie bedingte Sprünge (IF ... THEN) auf arithmetische Rechnungen zurückgeführt werden, obwohl sie so nahe an der Umgangssprache formuliert werden.

DER C 64 ALS RECHENMASCHINE

Arithmetik

Sie werden vielleicht verwundert sein, das Rechnen im üblichen Sinne als eine von vielen verschiedenen Tätigkeiten Ihres "Rechners" genannt zu finden. Tatsächlich werden im Computer auch Dinge wie Musik oder Logik auf dem Wege der Rechnung behandelt. Der Vorrat an rechnerischen Symbolen ist in BASIC ausgesprochen bescheiden: es gibt:

- die Grundrechnungsarten + - * / (nur mit /, nie mit :)
- die Potenzierung ↑
- die Quadratwurzel SQR()
- Logarithmus und Exponentialfunktion zur Basis e LOG() EXP()
- die Winkelfunktionen SIN() COS() TAN() im Bogenmaß,
- die Kreisfunktion arcustangens ATN()
- Absolutbetrag ABS() und Vorzeichen SGN() (mit den Werten -1,Ø und 1)
- die Abrundungsfunktion INT(), die auf die nächste nicht größere ganze Zahl abrundet.

Man kann damit aber eine Menge machen: der Cotangens ist natürlich 1/TAN(), der Logarithmus einer Zahl A zur Basis B ist LOG(A)/LOG(B); weitere Beispiele (vor allem arcussinus usw.) finden Sie im C 64-Handbuch und natürlich in mathematischen Formelsammlungen.

VORFAHRT EINGEBAUT

Klammern und Prioritäten

BASIC ist in der Schreibweise der Grundrechnungsarten sehr nahe bei der üblichen Mathematik. Das gilt auch für die Regel, daß "Punktrechnung vor Strichrechnung geht", d.h. daß a·b+c als (a·b)+c und nicht als a·(b+c) gedeutet wird. Das ist für den Computer zwar komplizierter, kommt aber den Gewohnheiten, die wir aus der Mathematikstunde kennen, sehr entgegen. Wenn keine Klammern gesetzt sind, gilt folgende Rangfolge (von innen nach außen, also von eng nach weit bzw. von früh nach spät in der Ausführung):

innen früh

1. Funktionen wie SIN() oder SGN() usw.
2. Minuszeichen als Vorzeichen
3. Potenzierung
4. * / (untereinander gleichberechtigt)
5. + - (Minus als Subtraktion, beide gleichberechtigt)
6. Vergleiche = <= >= <> (untereinander gleichber.)
7. logische Verneinung NOT
8. logische Konjunktion AND
9. logische Adjunktion: (nichtausschließendes) OR
10. Wertzuweisung = (LET-Anweisung)

außen spät

Mit runden Klammern (auch mehrfach ineinander, es müssen alle wieder geschlossen werden) kann man jede andere Rangfolge im Einzelfall festlegen. Stoßen gleichberechtigte Operationen ohne Klammern aufeinander, so wird von links nach rechts gerechnet, also: 3/4*12 ist 9.

Man sollte es mit den Klammern auch nicht übertreiben: das Stack (das ist ein Zählwerk, das unter anderem auch für Klammern zuständig ist) läuft sonst über. Es ist dann besser, Zwischenergebnisse als Variable einzuführen. Elegant, wenn auch nicht besonders übersichtlich ist es, die gleiche Variable der Reihe nach mit mehreren aufeinanderfolgenden Zwischenergebnissen zu füllen; übertriebenes Beispiel:

```
A=RND(Ø):A=A*754:A=INT(A):A=A+323
```

anstelle von A=INT(RND (Ø)*754+323).

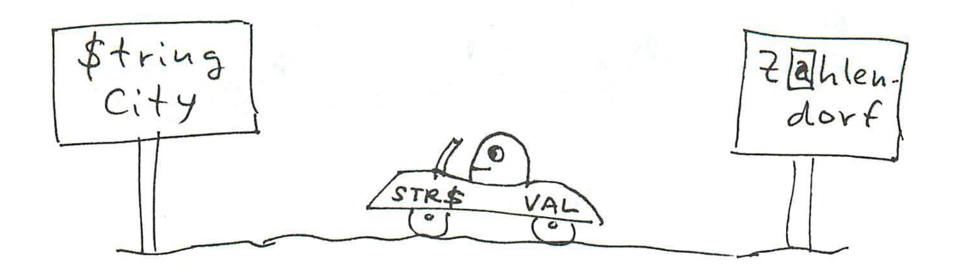
Beachten Sie bitte bei der Verwendung der logischen Verknüpfungen mit Zahlen, daß das Gleichheitszeichen (und die anderen Vergleichsoperationen) enger binden als AND, OR oder NOT: 3=(7 AND 3) ist -1 (weil zutreffend),
3= 7 AND 3 ist dagegen Ø, weil 3=7 unwahr ist und damit den Zahlenwert Ø hat. - Der Sinn der Sache liegt natürlich darin, daß Formulierungen wie die folgende ohne Klammer geschrieben werden können: IF A=B AND C=7 THEN ..., was bedeutet: IF (A=B) AND (C=7) THEN ... und nicht: IF(A=(7 AND C))=7 THEN. Leertasten können solche Dinge zwar für Menschen übersichtlich machen, für den Rechner zählen aber allein die vereinbarten Prioritäten und die Klammern.

Beachten Sie bitte die unterschiedliche Priorität des Gleichheitszeichens (das in BASIC leider doppelt belegt ist): Als Wertzuweisung bindet es weniger eng als in Vergleichen: Nach der Wertzuweisung A = 3 AND 6 ist A=2; die Bedingung A=3 AND 6 (etwa nach IF) hat für A=Ø den Wert Ø.

WERTZUWEISUNG
mit = und mit oder ohne LET

Format: b = a oder LET b = a
mit: Variable b und berechenbarem Ausdruck oder String a

Es gehört zu den größten Handicaps von BASIC, daß die Wertzuweisung mit dem symmetrischen Gleichheitszeichen geschrieben wird, obwohl beide Seiten denkbar verschiedene Bedeutungen haben: Links steht eine Variable, die einen (neuen) Wert bekommen soll, rechts steht dieser Wert (evtl. auch ein Ausdruck, der erst berechnet werden muß). In der Zuweisung A = A+B wird der Wert von A um den Wert B erhöht. Bei der Berechnung von A+B wird nämlich der alte Wert verwendet; nach der Ausführung der Anweisung hat die gleiche Variable A den neuen Wert. Sie sehen hier ganz deutlich, wie sehr es in BASIC auf die Reihenfolge von Anweisungen ankommt: A=B:B=A setzt beide Variablen auf den alten Wert von B, B=A:A=B umgekehrt auf den alten Wert von A.



EINMAL STRING UND ZURÜCK

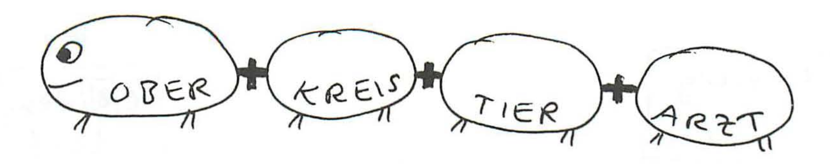
STR$() und VAL()

Was auf dem Bildschirm völlig gleich aussieht, nämlich wie eine Zahl, kann intern ganz verschieden gespeichert sein: als Zahl oder als String. Mit beiden Formen kann man ganz verschiedene Dinge tun: An das String 12.Ø54 kann man zum Beispiel vorlaufende Nullen anfügen oder hinten zwei Stellen abschneiden, mit der Zahl 12.Ø54 hingegen kann der C 64 rechnen. Zur Umwandlung zwischen beiden Formen gibt es die Funktionen STR$() und VAL(). Die erstere verwandelt eine Zahl in eine Zeichenkette, z.B. ist STR$(A) das String [37], wenn A zur Zeit gerade den Wert 37 hat. Denken Sie bitte daran, daß Zahlen ja im C 64 nicht in Dezimalziffern, sondern binär gespeichert sind. Bei einer positiven Zahl erscheint nach der Umwandlung in ein String links ein Leerzeichen anstelle des unterdrückten positiven Vorzeichens !

VAL() ist umgekehrt eine Funktion, die man auf Strings anwendet und die vom linken Ende an eine Zahl darin zu erkennen sucht. Dabei wird ein E in der Exponentialschreibweise richtig erkannt. Beispiele:

A$	VAL(A$)
"354 AEPFEL UND 54 BIRNEN"	354
"2.Ø76E-23 METER"	2.Ø76E-23
"F=3.7 NEWTON"	Ø

Im direkten Zugriff auf ein String muß dieses in Gänsefüßchen gesetzt werden: VAL("3 ORANGEN") gibt 3, VAL(3 ORANGEN) ist dagegen nicht zulässig (Type mismatch error).



BANDWÜRMER LÄNGER MACHEN

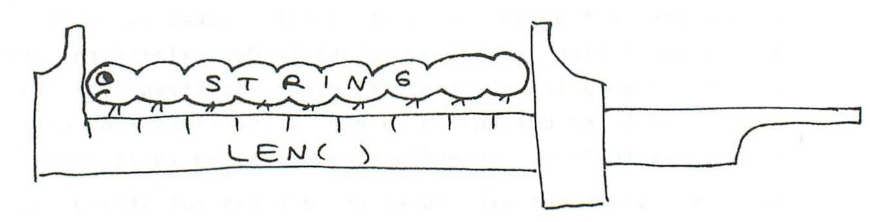
String-Verkettung mit +

Format: t1+t2 oder t1+t2+t3 etc.
mit Strings t1 und t2 etc.

Strings kann man aneinanderhängen mit dem Zeichen +, z.B.:

```
10 DATA OBER,KREIS,TIER,ARZT
20 FOR I=0 TO 3:READ A$(I):NEXT
30 FOR I=0 TO 3:B$=B$+A$(I):NEXT
40 PRINT B$
```

B$ ist vor diesem Programm (hoffentlich, d.h. wenn nichts anderes programmiert ist) das leere String "". Es wird also gedruckt: OBERKREISTIERARZT. Wenn Sie versehentlich die Zeile 30 so schreiben: FOR I=0 TO 3:B$=A$(I)+B$:NEXT, so erhalten Sie jedoch ARZTTIERKREISOBER

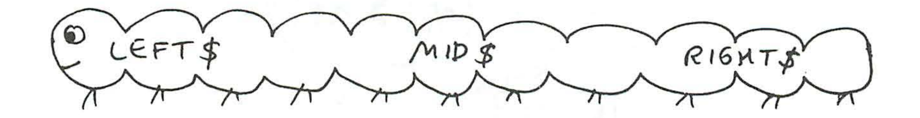


DAS MASSBAND

LEN()

Format: LEN(t)
mit String t, das Ergebnis ist eine nichtnegative ganze Zahl

Wenn A$ z.B. den Inhalt "DIES IST EIN TEXT" hat, so ist LEN(A$) die Zahl 17. MID$(A$,LEN(A$)-2,2) ist dann das Stück "EX". Zählen Sie es bitte nach und lassen Sie es den C 64 auch nachzählen.



VON KÖPFEN UND SCHWÄNZEN

LEFT$(), RIGHT$() und MID$()

Format: LEFT$(t,a) RIGHT$(t,a) MID$(t,n,a)
mit String t, Länge a des Teilstücks, Anfangsstelle n

Beispiele:
```
A$="DIES IST EIN TEXT"
PRINT LEFT$(A$,3)
```

Sie erhalten die Antwort: DIE .
Bei PRINT RIGHT$(A$,6) bekommen Sie: N TEXT ,
bei MID$(A$,8,4) schließlich: T EI
Praktisch ist auch oft: MID$(A$,5): es gibt:"IST EIN TEXT"

Und wenn Sie das 2. und 3. Zeichen von rechts haben wollen, aber nicht im voraus wissen, wie lang das String ist ?
Dafür gibt es MID$(A$,LEN(A$)-2,2)

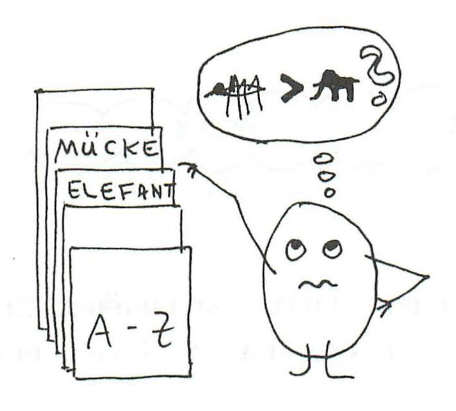
VON MÜCKEN UND ELEFANTEN

Vergleichen und Ordnen mit = < >

Format: t1=t2 t1> t2 t1< t2 t1<> t2
mit Strings t1,t2,t3
das Ergebnis ist Ø für unwahr oder -1 für wahr.

Mit dem Gleichheitszeichen = kann man Strings auf Gleichheit prüfen, z.B. in einem Quizprogramm (die richtige Antwort sei R$):

```
60 PRINT"WER WAR DER ERFINDER DES PHONOGRAPHEN "
70 INPUT A$:R$="EDISON"
80 IF A$=R$ THEN PRINT "RICHTIG":END
90 PRINT "LEIDER NOCH NICHT GANZ RICHTIG !"
```



Was kann aber die Relation "größer" bzw. "kleiner" bei Strings bedeuten ? Man könnte vielleicht an die Länge denken, aber dafür haben wir ja die Funktion LEN. Es wird etwas viel Besseres bewirkt: eine alphabetische Sortierung, genauer: eine Vergleichsabfrage der ASCII-Code-Nummern der Anfangszeichen, und wenn die gleich sind, der zweiten Zeichen usw. Da die normalen Buchstaben aufeinanderfolgende ASCII-Nummern haben, bedeutet das gerade eine Abfrage der Reihenfolge in einem Lexikon. Ein kleines (nicht sehr elegantes) Alphabetisierprogramm, das N Wörter sortiert, geht etwa so:

```
10 INPUT"ANZAHL DER WOERTER ";N:DIM A$(N)
20 FOR I=1 TO N:INPUT A$(I):NEXT
30 FOR I=1 TO N:FOR J=1 TO I
40 IF A$(I)<A$(J) THEN A$=A$(J):A$(J)=A$(I):A$(I)=A$
50 NEXT:NEXT
60 FOR I=1 TO N:PRINT A$(I):NEXT
```

Es werden also alle Kombinationen darauf abgefragt, ob sie in der falschen Reihenfolge stehen; falls ja, werden sie umgedreht (wobei die Variable A$ als Zwischenspeicher nötig ist, weil sonst beide hinterher gleich wären).
Daß man Zahlen ganz ähnlich nach ihrer Größe sortieren kann, versteht sich fast von selbst, ebenso, daß Klein- und Großbuchstaben nicht gemischt werden sollten (da sie im ASCII-Code in zwei Blocks liegen).



LOGISCH ODER ?

Wahrheitswerte und Binärzahlen

Daß 4 AND 6 als Ergebnis 4 hat, ist schon recht merkwürdig. Und fragen wir, was A=B sei, so erhalten wir die Antwort -1, wenn über A und B noch nichts bekannt ist, andernfalls aber auch eventuell eine Ø. Das Ganze ist bei näherem Hinsehen sehr vernünftig und sogar sehr nützlich.

Im vorigen Jahrhundert hat George Boole die Logik wie eine Art Algebra betrieben mit Rechenregeln für wahre und falsche Aussagen. Man nennt darum Zahlen, die zwischen "wahr" und "falsch", W und F unterscheiden, Boole-sche Variable, meist 1 und Ø, bei uns aber -1 und Ø (der Grund wird etwas später klar). Aussagen wie "A=B" oder "Heute ist Sonntag" sind manchmal wahr, manchmal auch falsch. Die Doppelaussage "A = B und heute ist Sonntag" kann auch wahr oder falsch sein, Hier gibt es aber eine einfache Regel: sie ist genau dann wahr, wenn beide Einzelaussagen (jede für sich) wahr ist, in allen anderen Fällen ist sie falsch. Allgemein:

(W und W) ist W	(W und F) ist F
(F und W) ist F	(F und F) ist F

Wie steht es nun mit der Aussage "A=B oder heute ist Sonntag"? In der Umgangssprache ist nicht ganz klar, ob das auch wahr ist, wenn beide Einzelaussagen wahr sind. Die Logik hat darum zwei verschiedene Wörter für "oder", ein ausschließendes und ein normales (nicht-ausschließendes). In BASIC kommt nur das normale vor, es heißt dort OR, das ausschließende kommt in

einigen anderen Programmiersprachen oder BASIC-Erweiterungen vor: In SIMON's BASIC heißt es EXOR, in Assembler meist XOR. Eine Aussage mit dem (nicht-ausschließenden) "oder" ist also immer wahr, wenn mindestens eine Teilausage wahr ist:

(W oder W)ist W	(W oder F)ist W
(F oder W)ist W	(F oder F)ist F

Probieren Sie das mit Ø und -1 aus:

```
PRINT Ø OR -1
```

Der Computer antwortet: -1. Auf diese Weise können Sie alle 8 Fälle aus den beiden Wahrheitstafeln (für AND und für OR) durchprobieren. Auch die Negation NOT ist sehr einfach:

```
PRINT NOT Ø
```

Antwort: -1, entsprechend umgekehrt.

Wenn nun irgendeine Abfrage wie A=B oder A=<C vorkommt, wird daraus einer der beiden Wahrheitswerte Ø (falsch) oder -1 (wahr) gemacht, also eine Boole-sche Variable belegt. Sie können sich nun überlegen, wie eine Bedingung wie IF A=7 AND B<-3 THEN ... bearbeitet wird.

Zur Sicherheit hier noch einmal die Wahrheitstafeln mit den Werten -1 (wahr) und Ø (falsch), wie sie in unserem BASIC auftreten:

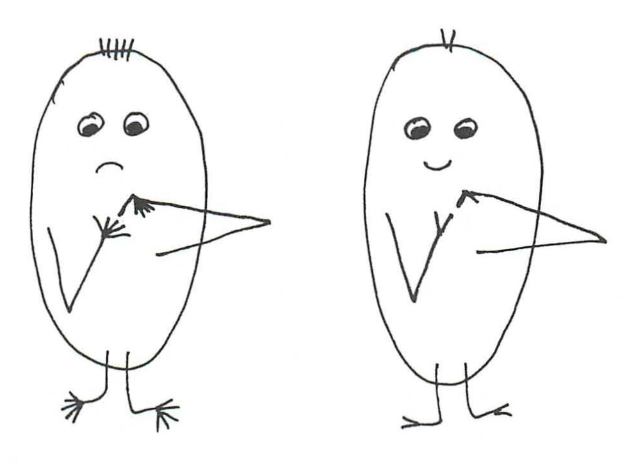
NOT Ø = -1
NOT -1 = Ø

(-1 AND -1)= -1 (Ø AND -1)=Ø	(-1 AND Ø)= Ø (Ø AND Ø)= Ø
(-1 OR -1)= -1 (Ø OR -1)= -1	(-1 OR Ø)= -1 (Ø OR Ø)= Ø

Damit ist noch nicht klar, warum die Wahrheit durch so eine seltsame Zahl wie -1 dargestellt wird, und was es zu bedeuten hat, wenn man die Boole-schen Operationen AND, OR und NOT auf andere Zahlen anwendet.

Dazu müssen wir uns die Binärdarstellung der Zahlen ansehen. Im Dezimalsystem haben wir 10 Ziffern und zählen:

Ø 1 2 3 4 5 6 7 8 9



Nun sind alle verbraucht, und wir müssen zweistellig weiterzählen: 1Ø 11 12 usw. Die linke 1 bedeutet darin etwas ganz anderes als eine rechts stehende, nämlich das Zehnfache. Die 1 in 1ØØ bedeutet sogar noch einmal das Zehnfache davon usw. Aber wozu braucht man überhaupt 10 verschiedene Ziffern? Der Computer kommt mit 2 aus (die anderen hat er nur, damit wir seine Ergebnisse in unserer gewohnten Weise ablesen und unsere Eingaben machen können). Zählen wir also mit nur den beiden Ziffern Ø und 1: Ø 1 1Ø 11 1ØØ 1Ø1 11Ø 111

In der "Binärzahl" 1Ø bedeutet die linke 1 nun keineswegs das Zehnfache einer gewöhnlichen 1, sondern nur das Zweifache, eine 1 an der dritten Stelle von rechts wieder davon das Doppelte, also eine Vier usw.

Was machen wir nun mit den negativen Zahlen, also wenn es um Zahlen geht, die noch kleiner als Ø sind ? Die Schulmathematik setzt die Folge ...3 2 1 Ø fort mit -1 -2 -3 usw. oder die Folge 0,03 0,02 0,01 0 mit -0,01 -0,02 usw. Das hat zwei Nachteile: Im Gegensatz zu allen anderen Zahlen fällt die Null mit ihrem Gegenstück zusammen, und die Rechenregeln sind durchaus nicht besonders einfach.

Betrachten wir nun ein Zählwerk, wie es z.B. am Kilometerzähler von Fahrzeugen oder auf Verbrauchszählern für Energie, Gas oder Wasser vorkommt: Jedes Rad bedeutet eine Dezimalstelle und trägt die Ziffern von 0 bis 9. Löst im Anzeigefeld die 0 die 9 ab, so wird die nächst-"höhere" Scheibe um eine Ziffer erhöht (in der Sprache des Kopfrechnens: Addition von Eins mit Übertrag, "Eins im Sinn"). Nun kann man ein Zählwerk auch rückwärts laufen lassen (die vorhin genannten Geräte lassen das leider nicht zu, es sei denn gewaltsam und mit betrügerischer Absicht). Löst nun die 9 die 0 im Anzeigefeld ab, so geht die nächst-"höhere" Scheibe um eine Ziffer rückwärts, und wenn die dabei von 0 auf 9 wechselt, die nächste auch noch usw. Bei einem sechsstelligen Dezimalzählwerk sieht das Rückwärtslaufen dann so aus:

000003
000002
000001
000000
999999
999998
999997 usw.

Sie können sich leicht überlegen, daß die Rechenregeln für die Addition und die Subtraktion mit dieser Schreibweise viel leichter sind als bei der Vorzeichenschreibweise. Der einzige Nachteil ist, daß man sich bei den negativen links unendlich viele Neunen denken muß, genau so wie es bei den positiven dort unendlich viele Nullen gibt. Man könnte die unendlich vielen Neunen symbolisch als $\overline{9}$ schreiben (analog zu der $\overline{3}$ in $1/3 = 0,\overline{3}$, die unendlich viele Dreien bedeutet). Daß man die unendlich vielen Nullen bei den normalen Zahlen einfach weglassen kann, liegt daran, daß es bei der gewöhnlichen Schreibung eindeutig ist, daß Leerstellen links Nullen bedeuten.

Bei einem Zählwerk ist nun die Anzahl der Stellen begrenzt, bei den Zahlen im Computer ebenfalls. Soll man nun 999993 als eine Zahl auffassen, die knapp unter der 0 liegt (also = -7 ist), oder als eine Zahl knapp unter einer Million ? Bei dem Zählwerk ist das eine offene Frage (falls es eins ist, das auch rückwärts laufen kann), und beim Computer muß das festgelegt werden. Wenn man es mit negativen und positiven Zahlen gleichermaßen zu tun hat, ist es beim Dezimalsystem sinnvoll, den Bereich zu halbieren: Ist die höchste Ziffer (also die ganz links stehende) eine von 0 bis 4, so soll die positive gemeint sein, ist es eine von 5 bis 9, die negative. Die Schreibweise mit Vorzeichen erhält man daraus, indem man 1 Million (bei sechsstelligen Zahlen) abzieht:

999994 wäre dann die vorzeichenlose Schreibweise
von -6.

Nun wenden wir uns wieder den Binärzahlen zu und zählen von 3 an 16-stellig rückwärts:

ØØØØØØØØØØØØØØ11	= dezimal 3
ØØØØØØØØØØØØØØ1Ø	2
ØØØØØØØØØØØØØØØ1	1
ØØØØØØØØØØØØØØØØ	Ø
1111111111111111	-1
111111111111111Ø	-2
11111111111111Ø1	-3

Das Zählen ist hier besonders einfach: Ganz rechts wird eine Ø mit einer 1 vertauscht oder umgekehrt. Beim Aufwärtszählen gibt es von 1 nach Ø einen Übertrag in die nächste Stelle, beim Abwärtszählen beim Wechsel von Ø nach 1. Wenn Sie die Spalten in der Zahlenkolonne ansehen, finden Sie, daß die Stelle ganz rechts (die Ø. Binärstelle) einfach zwischen Ø und 1 wechselt, auch über den Wechsel von positiven und negativen Zahlen hinweg. In der Stelle Nr. 1 findet der Wechsel jedes zweite Mal statt, in Nr. 2 jedes 4. Mal, in der Nr. 3 jedes achte usw.

Wir haben damit eine binäre Schreibweise für positive und negative ganze Zahlen, die sehr einfache Rechenregeln für die Addition und Subtraktion nach sich zieht. Der C 64 interpretiert diese Zahlen beim Umrechnen in Dezimalzahlen genau dann als negative, wenn die 15. Stelle (beachten Sie, daß die Numerierung der Stellen rechts mit Ø und nicht mit 1 anfängt !) eine Eins ist. Das sieht dann so aus:

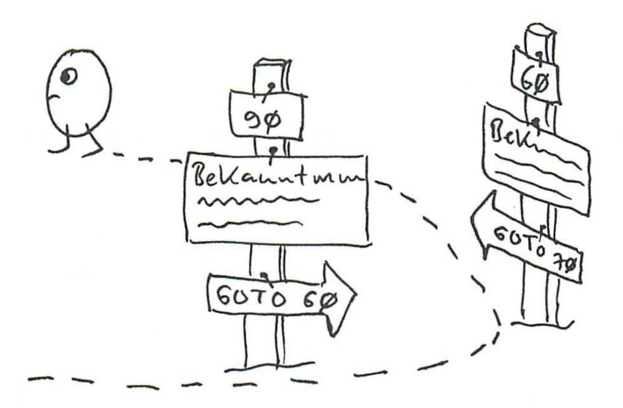
```
1ØØØØØØØØØØØØØØØ  =  -32768
1ØØØØØØØØØØØØØØ1  =  -32767
  ...
1111111111111111  =      -1
ØØØØØØØØØØØØØØØØ  =       Ø
ØØØØØØØØØØØØØØØ1  =       1
  ...
Ø11111111111111Ø  =   32767
Ø111111111111111  =   32367
```

Vertauscht man in einer solchen Zahl alle Nullen gegen Einsen und umgekehrt, so findet man eine neue Zahl, wobei die Summe aus beiden -1 ist. Genau diese Vertauschung macht die logische Operation NOT (logische Verneinung). NOT A ist also -A-1 (für jedes A). Damit ist auch klar, warum die Verneinung von Ø ausgerechnet (!) -1 ist und nicht +1. Wenn man die 1 in der 15. Binärstelle nicht zum Anlaß nimmt, die ganze Zahl als negativ aufzufassen, bekommt man eine um 2^{16} zu hohe Zahl. Diese Interpretation kommt beim C 64 auch vor, und zwar bei den Zeilennummern, die ja immer positiv sind, aber bis 63999 gehen dürfen.

Wird nun AND oder OR auf zwei Zahlen angewendet, so geschieht das einfach für jedes Bit einzeln:

37	ØØØØØØØØØØ1ØØ1Ø1
48	ØØØØØØØØØØ11ØØØØ
37 AND 48 =32	ØØØØØØØØØØ1ØØØØØ

37	ØØØØØØØØØØ1ØØ1Ø1
48	ØØØØØØØØØØ11ØØØØ
37 OR 48 =53	ØØØØØØØØØØ11Ø1Ø1



SCHNITZELJAGD

Der unbedingte Sprung mit GOTO

Format: GOTO n

mit einer (im Programm auftretenden !) Zeilennummer n

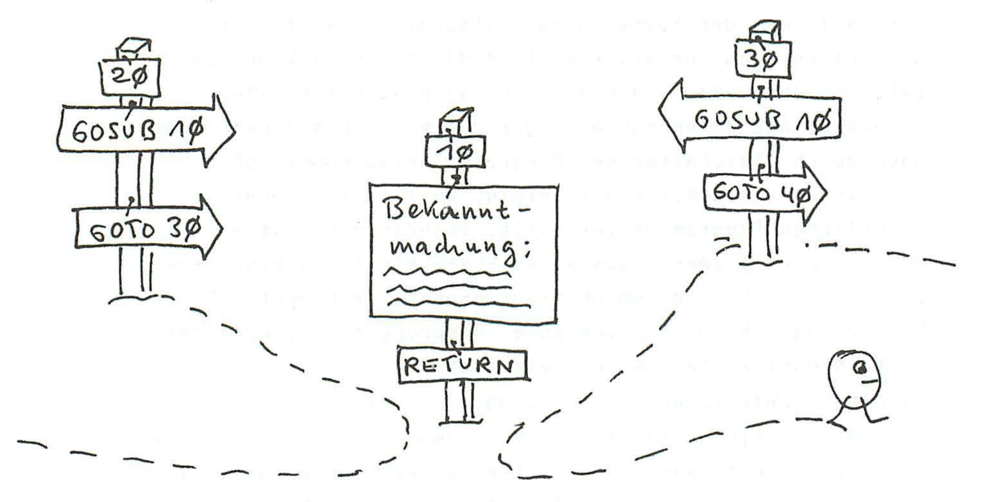
Die Zeile mit der Nummer n muß tatsächlich existieren, und sei es auch nur als Remark (REM). Mit GOTO kann man beliebig vor- und zurückspringen, also auch Schleifen erzeugen, aus denen man entweder durch IF-THEN-Anweisungen oder durch Abschalten des Programms herauskommt. GOTO gilt in der Informatik als unfein, weil es zu unübersichtlichen Programmen verleitet. Insbesondere ist es wenig empfehlenswert, aus einer FOR-Schleife in eine andere zu springen. Nach einem GOTO-Sprung hat der Computer keine Erinnerung mehr daran, von wo er gesprungen ist; er macht einfach dort weiter, wo er jetzt ist.

Eine Fangschleife der Form n GOTO n ist keineswegs so absurd, wie sie aussieht: Das Programm fängt sich in einer solchen Schleife wie in einem Kreisverkehr ohne Ausfahrt. Man kann diese (an sich unelegant aussehende) Form der Beendigung eines (sichtbaren) Programms sehr gut benutzen, wenn auf dem Bildschirm ein Bild mit PRINT oder mit POKE erzeugt worden ist: Ein END- oder STOP-Befehl würde eine störende READY-Mitteilung nach sich ziehen, die nicht selten das Bild ein Stück aus dem Bildschirm herausschiebt.

ABSTECHER
Unterprogramme mit GOSUB

Format: GOSUB n (im übergeordneten Programmteil)
RETURN (am Ende des untergeordneten Teils)
mit der Zeilennummer n, bei der der untergeordnete Teil beginnt

Die Anweisung GOSUB (was "go to subroutine" heißen soll) arbeitet ähnlich wie GOTO, markiert aber zusätzlich die Stelle, an der sie selbst steht,mit einem aktuellen Fähnchen (bildlich gesprochen), damit es beim nächsten RETURN wieder dort weitergehen kann. Das Unterprogramm ist nichts weiter als ein Programmstück, das mit der Anweisung RETURN endet und in das man mit GOSUB springt. Es muß vermieden werden, daß der Programmlauf anders dorthinein gerät, weil der Computer dann bei RETURN nicht wissen kann, wohin.



Steht ein Unterprogramm also hinter einem Hauptprogramm, so muß dieses mit END, STOP oder einer Fangschleife enden. Im Gegensatz zu anderen Programmiersprachen verwenden die Unterprogramme in BASIC die gleichen Variablen wie das Hauptprogramm. Für die "Übergabe" muß man also selbst sorgen.

Beispiel: Im Hauptprogramm tauchen Polarkoordinaten R1,W1 und R2,W2 auf, die jeweils umgerechnet werden müssen, bevor ein Punkt gezeichnet werden kann:

```
120 RR=R1:WW=W1:GOSUB 800: PLOT 160+X,100-Y,1
130 RR=R2:WW=W2:GOSUB 800: PLOT 160+X,100-Y,1
...
190 GOTO 190
...
800 X=RR*COS(W):Y=RR*SIN(WW):RETURN
```

Wie Sie an diesem Beispiel auch sehen, geht es bei GOSUB nach erfolgter Rückkehr in der gleichen Zeile weiter.

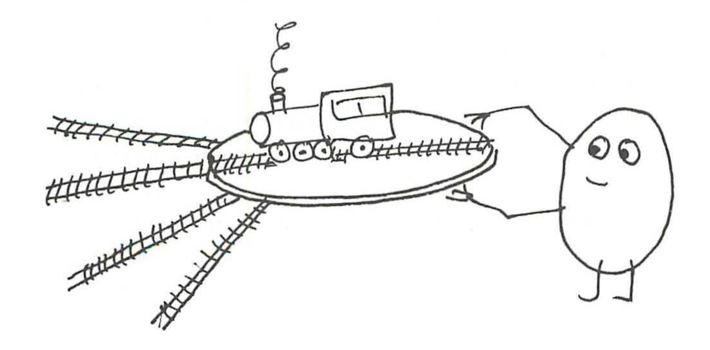
Mit GOSUB kann man auch in BASIC einigermaßen strukturiert programmieren, indem man einzelne Blöcke des Programms als Unterprogramme nacheinander oder auch ineinander verschachtelt mit GOSUB aufruft (natürlich ohne logische Purzelbäume !). In SIMON's BASIC gibt es wesentlich komfortablere Unterprogramm- und Sprunganweisungen, die Namen anstelle der Zeilennummern zur Adressierung verwenden und lokale Variable gestatten (PROC, CALL, EXEC, vgl. Handbuch zu SIMON's BASIC).

DER LOGISCHE VERSCHIEBEBAHNHOF

Der Sprungverteiler ON

Format: ON m GOTO n1,n2,n3 usw.
ON m GOSUB n1,n2,n3 usw.

mit einer Zahlenvariablen m und vorhandenen Zeilennummern n1,n2 usw



Hier stehen hinter GOTO bzw. hinter GOSUB gleich mehrere Zeilennummern, durch Kommata getrennt. Zu welcher gesprungen wird, hängt von der Zahl m ab: Ist sie (notfalls nach Abrunden) =1, so geht es zur ersten Adresse, bei 2 zur zweiten usw. Ist m kleiner als 1 oder (auch nach Abrunden) größer als die Zahl der Zeilennummern, so geht es mit der nächsten Zeile weiter. Oft möchte man einen Sprung von dem Vorzeichen (SGN) einer Zahl abhängig machen:

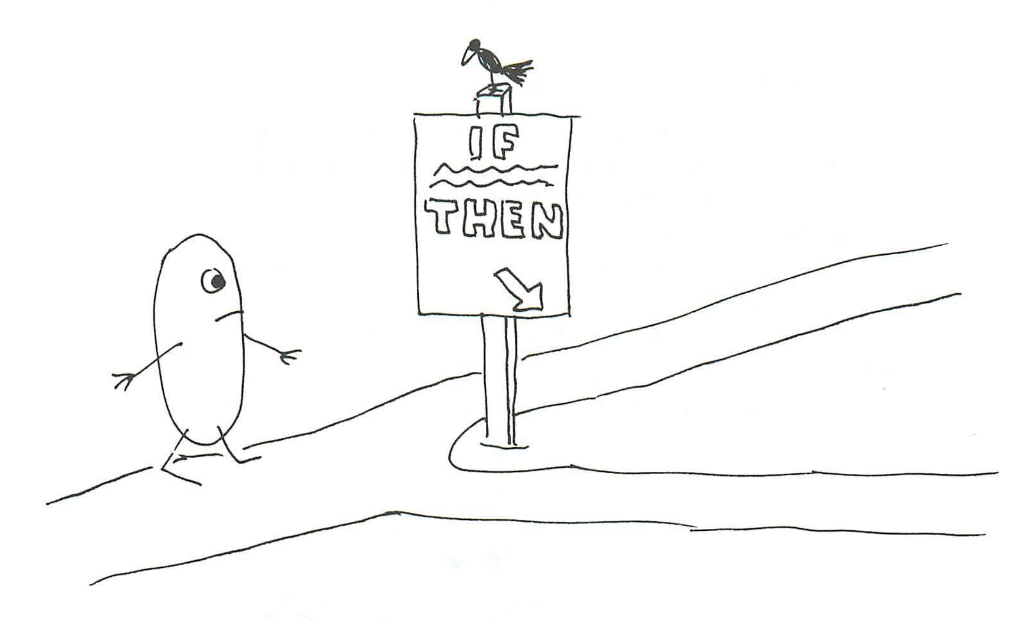
```
100 INPUT A
110 ON 2+SGN(A) GOTO 120,130,140
120 PRINT"A IST NEGATIV":END
130 PRINT"A IST NULL":END
140 PRINT"A IST POSITIV":END
```

SGN hat nämlich entweder den Wert -1 oder Ø oder +1.

IM FALLE EINES FALLES
Bedingte Anweisungen mit IF

Format: IF b THEN a
mit einer Bedingung oder Zahlenvariablen b und einer Anweisung(-sfolge) a (worin die anweisungen mit Doppelpunkten getrennt sind)

Was zwischen IF und THEN steht, wird in eine Zahl umgewandelt (wenn es nicht schon von vornherein eine ist), und zwar wird aus einer nicht erfüllten Bedingung (z.B. A=B, wenn A gerade den Wert 7 und B den Wert 8.3245 hat) eine Ø, aus einer zutreffenden (z.B. A=B, bevor diese irgendwelche Werte zugewiesen bekommen haben, also noch beide Ø sind) eine -1 gemacht. Ist das Ergebnis von b ungleich Ø (z.B. bei der zutreffenden Bedingung), werden die Anweisungen hinter THEN ausgeführt; ist es aber =Ø , so geht es in der nächsten Zeile weiter (nicht etwa mit der nächsten Anweisung: Alle Anweisungen hinter THEN in der gleichen Zeile hängen von der Bedingung ab !).



Muß von mehreren Bedingungen nur eine erfüllt sein, kann man sie mit OR koppeln:

```
IF A=4 OR B=7 THEN PRINT A$
```

Müssen mehrere Bedingungen gleichzeitig erfüllt sein, kann man sie entweder mit AND koppeln, oder die IF-Anweisungen schachteln:

```
IF A=Ø THEN IF B=5 THEN PRINT"A=Ø,B=5"
```

Wenn man die besonders unwahrscheinlichen Bedingungen voranstellt, kann man damit die Rechenzeit verkürzen (denn dann brauchen die anderen nicht mehr bearbeitet zu werden, falls die erste nicht zutrifft).

Fragen Sie nicht auf Gleichheit ab, auch nicht, wenn die Zahlen nach Ihrer Ansicht ganz sein sollten:

```
1Ø IF 9↑2=9*9 THEN PRINT"OK":END
11 PRINT"NANU ?"
```

Zur Erklärung: Die gleiche Rechnung wird hier auf zwei Arten formuliert und auch auf zwei Arten gerechnet; leider stimmen die Ergebnisse nur sehr genau, aber nicht ganz genau überein.

Machen Sie es lieber so:

```
10 IF ABS(A-127)<1E-6 THEN PRINT"A IST ETWA =127"
```

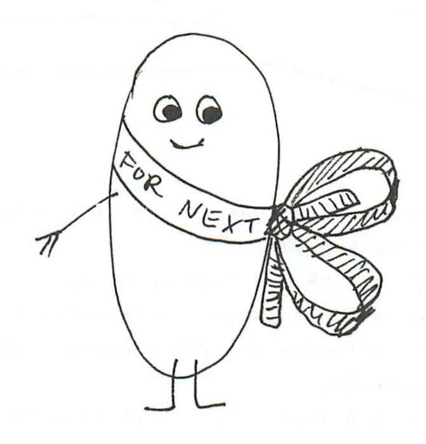
Wenn Sie Bedingungen zwischen IF und THEN schreiben, können Sie die ganze Anweisung wie in der (englischen) Umgangssprache verstehen. Mit Zahlen geht es manchmal kürzer:

```
20 IF A THEN PRINT "A IST UNGLEICH 0"
```

Sie können sogar ganze Rechenoperationen ohne IF und THEN abhängig machen:

```
10 INPUT A:PRINT 13-(A=>3)*3:GOTO 10
```

Der Ausdruck in der Klammer ist -1, wenn A größer oder = 3 ist, sonst Ø. Es wird also entweder 13 oder 16 ausgedruckt.



DAS GANZE GLEICH MEHRMALS

Schleifen mit FOR und NEXT

Format: FOR i = a TO e   oder   FOR i = a TO e STEP s
           NEXT   oder   NEXT i

Alle Anweisungen, die zwischen der FOR-Anweisung und NEXT stehen, werden mindestens einmal durchlaufen. Danach wird der Laufwert i, der mit a startet, jeweils um 1 bzw. um s (was auch negativ sein darf) erhöht. Wird der Endwert e nicht erreicht oder überschritten, so wird die Schleife erneut durchlaufen. Alle genannten Zahlen brauchen nicht festgelegt zu sein, sondern können sich im Programm als Variable

ergeben; der Laufindex i kann sogar in der Schleife durch andere Anweisungen abgeändert werden (was aber nicht unbedingt zur Übersichtlichkeit beiträgt). Man kann auch mehrere Schleifen ineinander verschachteln:

```
10 FOR J=1 TO 10:FOR K=1 TO 10
20 PRINT J;"*";K;"=";J*K
30 NEXT:NEXT
```

Dabei ist darauf zu achten, daß die Laufwerte i (im Beispiel J und K) dann verschieden sind (das wird leicht bei der Benutzung von Unterprogrammen übersehen).
Oft schreibt man eine ganze Schleife mit Inhalt in eine einzige Zeile. Das geht nicht, wenn im Innern z.B. IF...THEN steht: Die NEXT-Anweisung darf nicht von der IF-Bedingung abhängig gemacht werden.

Oft soll eine Schleife nur durchlaufen werden, wenn a kleiner oder gleich e+s ist. In manchen Fällen ist das durchaus ungewiß. Man muß dann vor die Schleife einen bedingten Sprung (mit IF a>e+s THEN GOTO n, wobei n die nächste Zeilennummer hinter NEXT ist) setzen, der die ganze Schleife überspringt.

WO BIN ICH ?

Silbentrennung mit POS()

Manche Funktionen sind nur für sehr wenige Dinge nützlich, dann aber sehr. Dazu gehört POS(0), womit einfach die Nummer der Spalte abgefragt wird, in der gerade geschrieben wird. Wenn Sie eine lange Liste von Wörtern schreiben lassen, kann dies in laufenden Zeilen geschehen: Erst wenn eine Zeile fast voll ist, soll es in der nächsten weitergehen. Das kann man so machen (die A$(i) seien mit Wörtern aufgefüllt, die nicht länger als 8 Zeichen seien):

```
70 FOR I=1 TO 50
80 PRINT A$(I)+" ";:IF POS(0)>31 THEN PRINT
90 NEXT
```

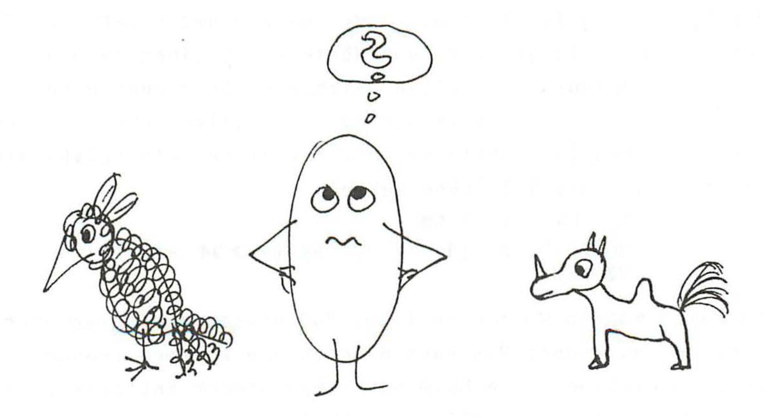
Nun sind manche Wörter so lang, daß dieses Verfahren nicht mehr gut aussieht: Man kann dann in den Wörtern Trennungszeichen einbauen, die beim Schreiben wieder entfernt werden (dazu brauchen Sie MID$() und LEN() in einer Schleife):

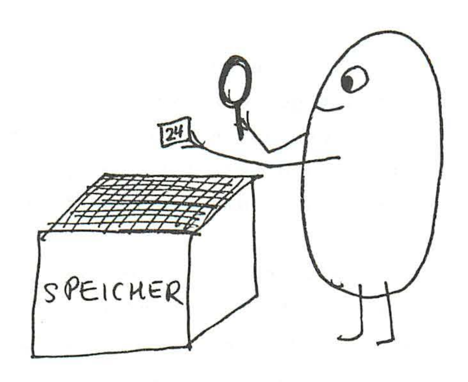
Bei Leerzeichen oder diesen Trennungszeichen wird die POS abgefragt und die Zeile gegebenenfalls gewechselt; am Schluß der Zeile bleibt das Trennungszeichen natürlich stehen.

Sie können auch versuchen, die Regeln des DUDEN über die Silbentrennung in BASIC umzuschreiben (statt die möglichen Trennungsstellen bei den einzelnen Wörtern vorzugeben). Das ist natürlich viel schwieriger, und es wird nie so weit gehen, daß der Computer weiß, wie er STAUBECKEN trennen soll, da er nicht wissen kann, ob es Becken zum Stauen oder Ecken voller Staub sind.

```
7 REM SILBENTRENNUNG (DER LUSTIGE ZOO)
8 :
10 DATA AMEI-SEN-,WAN-DER-,BREIT-MAUL-,ZOT-TEL-,RIE-SEN-
20 DATA KROE-TE,EU-LE,AF-FE,NAS-HORN,KRO-KO-DIL
30 FORI=0 TO 4:READ A$(I):NEXT
40 FORI=0 TO 4:READ B$(I):NEXT
50 C$=A$(RND(1)*5)+B$(RND(1)*5)
60 FOR I=1 TO LEN(C$)
70 D$=MID$(C$,I,1)
80 IF D$="-" AND POS(0)<36 THEN NEXT
90 IF D$="-" THEN PRINT D$:PRINT:NEXT
100 PRINTD$;
110 NEXT
120 FOR J=0 TO 2000:NEXT
130 PRINT"  ";:IF POS(0)>36 THEN PRINT:PRINT
140 GOTO 50
```

Bauen Sie dieses Programm zu einem großen Wunderzoo aus !





DIE GANZ DIREKTE TOUR
Schreiben und Lesen im Speicher mit POKE und PEEK()

Format: PEEK(n)
POKE n,b
mit Speicherplatznummer n (Ø bis 65535) und
Byte b (Ø bis 255)

Fast alle BASIC-Anweisungen arbeiten mit Variablen, von denen der Computer weiß, wo er ihre Werte notiert hat. Man kann aber auch direkt zugreifen. Die Abfrage erfolgt mit der Funktion PEEK() von der Speicherplatznummer, die eine ganze Zahl von Ø bis 65535 sein muß (notfalls wird automatisch abgerundet), z.B.:

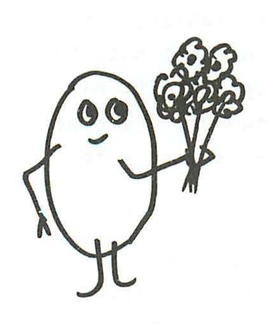
```
PRINT PEEK(2Ø3)
```

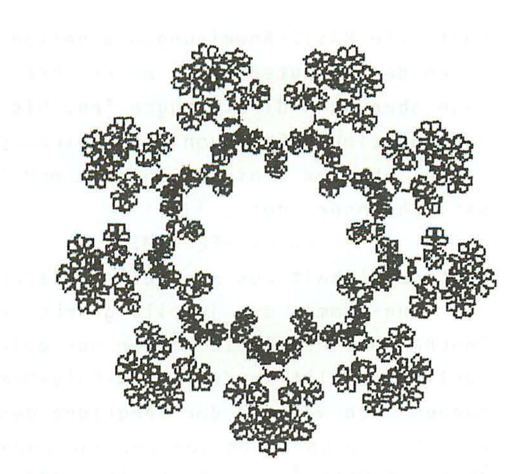
gibt den Inhalt des speziellen Platzes Nr. 2Ø3 aus, der die Kenn-Nummer der jeweils gedrückten Taste enthält. Manche Speicherplätze können nur gelesen, aber nicht willkürlich gefüllt werden: Read-Only-Memory = ROM. Andere können auch während der Benutzung des Computers mit neuen Inhalten beschrieben werden; man nennt sie RAM = Random-Access-Memory (was jedoch etwas anderes bedeutet: daß man sie in beliebiger Reihenfolge ansteuern kann). Mit der BASIC-Anweisung POKE kann man im RAM schreiben, z.B.

```
POKE 65Ø,128
```

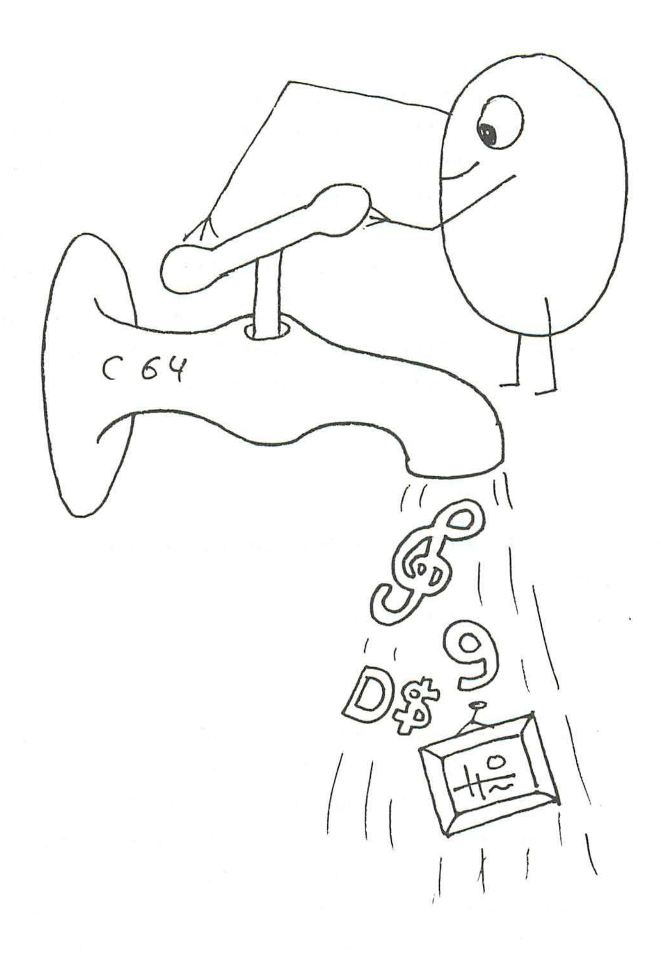
Dieser Speicherplatz wird bei gewissen Routinen abgefragt, und wenn Sie den Inhalt geändert haben, so beeinflussen Sie damit den Verlauf dieser Routinen,so als hätten Sie eine Weiche bei einer Eisenbahn gestellt. Im speziellen Fall dieses Beispiels geht es um die Dauerfunktion der Tasten: vgl. Seite 37
Bei vielen anderen Speicherplätzen kann man sich das ähnlich vorstellen wie das Stellen einer Weiche.

Viele trickreiche POKE-Adressen wirken wie Zauberei, andererseits ist es leichtsinnig, wahllos mit POKE in dem Speicher "herumzustochern", wenn im Speicher ein noch nicht extern abgespeichertes Programm steht. Die Gefahr ist sehr groß, daß der Rechner "abstürzt" und nur durch Aus- und Wieder-Einschalten ans Laufen gebracht werden kann. Man sollte daher Programme mit POKE immer erst SAVEn und dann starten. Wird eine POKE-Adresse im Programm berechnet oder gar gewürfelt (etwa bei der Grafik), so muß im Programm dafür gesorgt werden, daß der zulässige Adressenbereich nicht überschritten wird.





```
10 REM BLUME MIT SIMON
11 :
20 HIRES 1,0
30 FOR W=0 TO 1024*π STEP.3
40 X=160+65*SIN(W/512)-23*SIN(W/64)+7*SIN(W/8)-3*SIN(W)
50 Y=100+65*COS(W/512)+23*COS(W/64)+7*COS(W/8)+3*COS(W)
60 PLOTX,Y,1:NEXT
70 GOTO 70
```



DIE AUSGABE VON DATEN

Damit der C 64 nicht für sich behält, was er so berechnet hat, gibt es verschiedene Ausgaben:

1. über den Bildschirm: Zahlen, Texte und Bilder, alles auch bunt
2. über den Lautsprecher des Fernsehgerätes oder einen HiFi-Verstärker mit Lautsprecher
3. über einen angeschlossenen Drucker

4. an einen externen Datenträger (Tonbandcassette, Diskette), zwecks späterer Eingabe in den Rechner zurück (das betrifft auch die Programme selbst)
5. über andere externe Geräte, die ihrerseits Geräte wie Maschinen oder Roboter steuern.

In diesem Buch geht es hauptsächlich um die zuerst genannten Möglichkeiten, vor allem um die Grafik (im Zusammenhang damit dann allerdings auch um Drucker und externe Träger).

DER C 64 ALS PROGRAMMIERTER RECHNER

Zahlenausgaben

Die Ausgabe von Zahlen mit PRINT X erfolgt je nach dem Wert der Variablen X als Ganzzahl, Fließpunktzahl (z.B. 3.76) oder Fließpunktzahl mit Exponenten (z.B. 6.367E-23). Leider ist eine normgerechte Tabellierung damit sehr umständlich: Es gibt hier keine Format-Anweisung, die dafür sorgen würde, daß alle Dezimalpunkte untereinander, also in der gleichen Spalte erscheinen. Man kann zwar allerhand String-Operationen auf die Zahlen anwenden, muß aber die mit E geschriebenen Zahlen dabei besonders behandeln. Relativ einfach geht es mit Integer-Zahlen: z.B. PRINT RIGHT$("   "+STR$(A%),6)
Mehrspaltiges Ausgeben von Zahlen ist nur bei Integer-Zahlen sinnvoll, da die Anweisung PRINT X1,X2 die beiden Zahlen auf vier Spalten mit je einem Viertel der Bildschirmbreite verteilt. Fließpunktzahlen sind jedoch dafür oft zu lang, und es gibt dann ein häßliches Überlaufen in die nächste Zeile. Kurze Zahlen, insbesondere mit erzwungener Ganzzahligkeit (Denken Sie daran, daß auch Zahlen, die theoretisch ganz sein sollten,

aufgrund von ungenauer Rechnung viele Nachpunktstellen haben können !) kann man mit TAB( ) tabellieren, z.B.:

```
80 PRINT X1;TAB(7)X2;TAB(14)X3
```

(noch besser mit der oben erwähnten Stringsbehandlung, damit sie nicht links- sondern rechtsbündig erscheinen).

DER C 64 ALS DIALOGPARTNER

Textausgabe

Während bei Zahlen in PRINT-Anweisungen stets ein Leerzeichen nachläuft und ein positives Vorzeichen als Leerzeichen erscheint, werden bei der Verwendung des Semikolons zwischen Stringvariablen in der PRINT-Anweisung keine Zwischenräume gelassen: PRINT "OBER";"KREIS";"TIER";"ARZT"
liefert also: OBERKREISTIERARZT

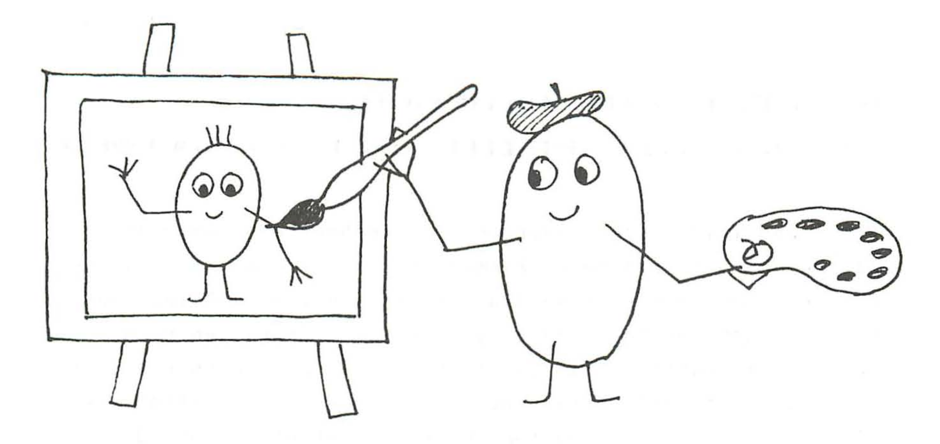
Farb- und Reverse-Umschaltungen sowie Cursor-, HOME- und CLR-HOME-Schritte können als eigene Strings (direkt oder in Form von Stringvariablen) und als Bestandteile von Strings in PRINT-Anweisungen verwendet werden. Damit kann man z.B. in einem laufenden Text Hochzahlen oder unten angehängte Indexzahlen schreiben, z.B. Na_2SO_4 oder $a^2+b^2=c^2$. Sie können auch Längenzeichen in lateinischen Grammatikprogrammen verwenden (Cursor auf, _ , Cursor ab, Cursor links, Vokal) oder einzelne Wörter oder Wortbestandteile in anderen Farben schreiben als den übrigen Text. Zu einem schönen Schriftbild gehört der großzügige Gebrauch von Leerzeilen (am einfachsten mit "Cursor ab" am Beginn der neuen Zeile), vor allem wenn man Großbuchstaben verwendet. Die Beachtung der Silbentrennung an den Zeilenenden sollte Ehrensache sein. Sie ist auch dann möglich, wenn innerhalb einer Zeile mehrere PRINT-Anweisungen (die dann mit dem Semikolon schließen müssen) aufeinanderfolgen, ohne daß es immer die gleichen sind. Dazu gibt es die Funktion POS().

FARBEN

Auf dem C 64 stehen 16 Farben zur Verfügung, sie sind von Ø bis 15 durchnumeriert. Diese Nummern werden benutzt, wenn Farben mit POKE oder mit den SIMON-Anweisungen COLOUR, HIRES, HI COL, LOW COL, MULTI gesetzt werden. In Verbindung mit PRINT werden ihnen ASCII-Nummern zugeordnet (die sonst eine genormte Numerierung von Buchstaben und anderen Zeichen bilden). Die einfachste Art, Farben (allerdings nur für Texte, nicht Rahmen, Hintergrund oder Feingrafik; Tastaturgrafik wird dabei wie Text behandelt) einzugeben, geschieht innerhalb von Strings mit den Ziffertasten 1 bis 8, zusammen mit CTRL bzw. mit der Commodore-Taste C=. Die Farben Nr. Ø bis 7 erreicht man mit Tasten 1 bis 8 und CTRL, Nr. 8 bis 15 mit Tasten 1 bis 8 und C=.

Wenn man Rücksicht darauf nimmt, daß ein Programm auch einmal auf einem Schwarzweiß-Fernsehgerät benutzt wird, sind die Helligkeiten zu beachten; es bleiben dann noch etwa 5 deutlich verschiedene unbunte Farben übrig:

Farbe	Nr.	ASC	Tasten	Unbuntdarstellung
Schwarz	Ø	144	CTRL 1	Schwarz
Rot	2	28	CTRL 3	Dunkelgrau
Blau	6	31	CTRL 7	"
Braun	9	149	C= 1	"
Dunkelgrau	11	151	C= 4	"
Grün	5	3Ø	CTRL 6	Mittelgrau
Purpur (Magenta)	4	156	CTRL 5	"
Orange	8	129	C= 2	"
Hellrot	1Ø	15Ø	C= 3	"
Mittelgrau	12	152	C= 5	"
Hellblau	14	154	C= 7	"
Cyan	3	159	CTRL 4	Hellgrau
Gelb	7	158	CTRL 8	"
Hellgrün	13	153	C= 6	"
Hellgrau	15	155	C= 8	"
Weiß	1	5	CTRL 2	Weiß



Nach Tasten angeordnet, sieht das so aus:

Taste	mit CTRL		mit C=	
1	Schwarz		Orange	
2	Weiß		Braun	
3	Rot	<--------->	Hellrot	
4	Cyan		Dunkelgrau	<---
5	Purpur		Mittelgrau	<---
6	Grün	<--------->	Hellgrün	
7	Blau	<--------->	Hellblau	
8	Gelb		Hellgrau	<-----

Leider stehen nur die ersten 8 Farben in englischen Abkürzungen auf den Tasten; darum können Ihnen die Striche in dieser Tabelle als Merkhilfe dienen: Rot, Grün und Blau haben jeweils noch eine helle Variante, die drei Graustufen sind von Dunkel nach Hell eingefügt.

Und nun zum Einsatz der Farben. Zunächst wollen wir den Bildschirmhintergrund (Papierfarbe sozusagen) und den Rahmen festlegen. Das geschieht in zwei bestimmten Speicherplätzen:

```
10 FOR F=0 TO 15:POKE 53280,F
20 FOR I=0 TO 1000:NEXT:NEXT
```

Dieses Programm zeigt Ihnen alle Rahmenfarben. Mit dem Platz Nr. 53281 geschieht das gleiche mit dem Hintergrund. In SIMON's BASIC kann man das mit COLOUR machen:

Die Syntax ist: COLOUR fr,fh für die Rahmenfarbe fr und die Hintergrundfarbe fh, jeweils Ø,1,2,...,14,15.

SCHNELL UND EINFACH

Bilder mit PRINT und Normalgrafik

Wer einen C 64 (oder einen anderen Rechner von Commodore oder auch einigen anderen Firmen) hat, sollte wirklich keine Grafiken aus so primitiven Buchstaben wie I oder O zusammenbasteln. Schließlich sind ja auf den meisten Tasten noch zwei Grafikbausteine mitgeliefert, und alle Zeichen kann man "revers" (d.h. mit Vertauschung von Papier- und Tintenfarbe, wie es anschaulich genannt werden kann) setzen und obendrein in 16 verschiedenen (Tinten-)Farben auf 16 verschiedenen (Papier-)Farben. Sie haben sicher schon im direkten Zugriff auf diese Weise Bilder zusammengebaut. Um dasselbe auch in einem Programm zu tun, muß noch einmal an den Gänsefußmodus erinnert werden:

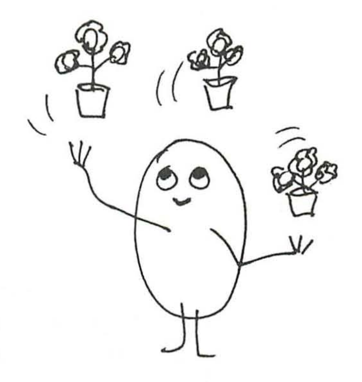
Es gibt im C 64 eine Stelle, die festhält, ob seit dem letzten Drücken von RETURN das Gänsefüßchen " eine ungerade oder eine gerade Anzahl oft gedrückt wurde (das Löschen von Gänsefüßchen auf dem Bildschirm hat darauf keinen Einfluß !). Bei "gerade" werden Cursor- und Farbeingaben (einschließlich Reverse On/Off und Home/Clear-Home) direkt ausgeführt, d.h. z.B. daß die Schrift ihre Farbe ändert. Das hat aber beim Schreiben eines Programms keinen Einfluß auf die spätere Ausführung. Ist aber das Gänsefüßchen " einmal (oder dreimal usw.) gedrückt worden, so erscheinen auf dem Bildschirm seltsame Zeichen, aber durchaus in der alten Farbe. Sie bleiben im Programm stehen und bewirken später bei der Ausführung, daß dann die Farbe wechselt oder daß dann der Cursor z.B. nach oben springt oder nach HOME.

Am besten üben Sie zunächst im direkten Modus, wie man mit den Reverse-, Farb- und Cursor-Steuerungen Bilder zeichnet, und geben dann später die gleichen Anschläge hinter einem Gänsefüßchen in einer PRINTanweisung ein (natürlich dann mit einer Zeilennummer).

Sie können auch einzelne kleine Bilder als Stringvariable zuweisen, z.B.

```
100 REM  BLUMENTOEPFE
110 :
120 A$="[illegible]"
130 PRINT"[illegible]";:FOR I=0 TO 6
140 FOR J=0 TO 8
150 PRINT A$"[illegible]";:NEXT
160 PRINT:PRINT"[illegible]  ";:NEXT
170 GOTO 170
```

Steuerzeichen
vgl. Anhang S.238



Das bunte Bild kann dann wie ein Textwort mit PRINT an eine gewünschte Stelle "geschrieben" werden: 7Ø PRINT A$(3). Die Stelle, an der es dann erscheint, richtet sich nach der Position des Cursors zu dem jeweiligen Zeitpunkt im Programmlauf. Zur "relativen" Steuerung des Ortes eignen sich entsprechend viele Cursortasten zwischen Gänsefüßchen für die jeweiligen Richtungen. Beachten Sie dabei, daß auch der Sprung nach rechts rückgängig gemacht werden muß, der nach jedem Schreiben stattfindet, und daß Sie natürlich jede PRINT-Anweisung mit dem Semikolon abschließen müssen, wenn der Cursor nicht an den Beginn der nächsten Zeile springen soll.

Für eine "absolute" Steuerung des Ortes (z.B. wenn Sie die Bilder in einem Bilder-Memory-Spiel verwenden, wo sie abwechselnd an verschiedenen Plätzen erscheinen müssen) gibt es die Taste HOME mit nachfolgenden abgezählten (oder in FOR-Next-Schleifen abgepackten) Cursorschritten nach rechts und nach unten.
Beachten Sie auch den Unterschied zwischen der Leertaste und der Cursor-Steuerung nach rechts: Die Leertaste radiert zugleich aus !

Noch ein Trick, der das Flimmern vermindert: Soll sich ein Bild aus mehreren Zeichen Höhe und Breite schrittweise über den Bildschirm bewegen, so ist es günstig, die relative Bewegung zu verwenden und nicht das ganze Bild mit Leertasten zu löschen, sondern nur die Teile, die nicht ohnehin vom Bild in der neuen Position ersetzt werden.

Die relative Positionierung hat leider einen Nachteil: Man kann sie nur auf einen Gegenstand anwenden, da es für den Cursor nur eine Position zu jedem Zeitpunkt gibt. Bewegen sich zwei Objekte unabhängig voneinander über den Bildschirm, so muß man sie entweder "absolut" (d.h. jedesmal von der HOME-Ecke aus) lenken oder aber (zumindest für das zweite und alle weiteren) auf die Grafik mit POKE ausweichen.

MANCHMAL BESSER:
Bilder mit POKE und Normalgrafik

Dieses Verfahren leistet an fertigen Bildern das gleiche wie das vorige, ist im allgemeinen langsamer und bei Bildern aus mehreren Zeichen umständlicher. Interessant ist es trotzdem: Zum einen, weil es die Organisation des Bildschirms offenlegt, und zum anderen wegen des Überganges zu feineren Methoden, besonders zur Viertelpunktgrafik und zu Histogrammen.

Der HOME-Position sind zwei verschiedene Speicherplätze zugewiesen, und zwar:

für die Form: 1Ø24
für die Farbe 55296

Die jeweils folgenden Speicherplätze betreffen die anderen Plätze auf dem Bildschirm, und zwar zeilenweise von links nach rechts, wie die Buchstaben auf einer Buchseite, aber nur mit 4Ø Nummern pro Zeile und 25 Zeilen (vgl. Abbildung im Anhang Seite 172). Wenn Sie nun ein bestimmtes Zeichen eintragen möchten, müssen Sie aus Zeilen- und Spalten-Nummer die Speicherplatzadressen berechnen und in der Liste "Bildschirm-Code" die Nummer für die Zeichen-Form nachsehen und natürlich den Farbcode. Um z.B. den vollen Kreis (Nr. 81) in die Spalte X und Zeile Y (gezählt von links oben, jeweils mit Ø als Anfang) zu setzen, z.B. in der Farbe Rot (Nr.2), brauchen Sie die Anweisungen:

```
10 POKE 1Ø24+X+Y*4Ø,81:POKE 55296+X+Y*4Ø,2
```

(Das Ausrechnen überlassen wir also dem Computer: wozu kann er schließlich rechnen !)
Achten Sie bitte bei allen POKE-Anweisungen darauf, daß die Adressen im zulässigen Bereich liegen ! Es muß dafür gesorgt werden, daß X oder Y nicht negativ oder zu groß werden (notfalls durch IF-Abfragen). Vergessen Sie auch die Konstanten 1Ø24 bzw. 55296 nicht, oder wenn Sie sie zu Variablen machen, achten Sie darauf, daß sie nicht im entscheidenden Augenblick den Wert Null haben: POKE-Anweisungen mit niedrigen Adressen lassen den Computer oft abstürzen. Dann hilft nur noch der Netzschalter, und das Programm muß neu eingegeben werden (Faustregel: Programme mit POKE erst SAVEn, dann starten !). Bedenken Sie bitte, daß POKE ein ziemlich direkter Eingriff in das Innenleben des Computers ist und für die Software so

ähnlich ist, wie für die Hardware ein Anbringen irgendeiner Lötverbindung. Niemand wird das blind und auf Verdacht machen ! Aber Sie sollen vor POKE auch nicht zuviel Angst bekommen: Mehr als Ihr eigenes Programm können Sie damit nicht kaputtmachen !

GAR NICHT SO GROB

Feine Balkengrafik 25 x 320

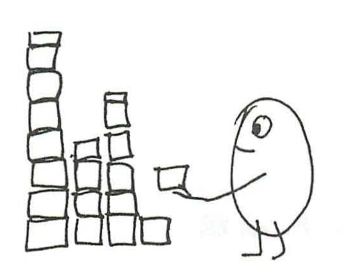
Um Größen zu veranschaulichen, kann man sehr gut senkrechte oder waagerechte Säulen zeichnen. Die von den Tasten her erreichbaren Grafikzeichen erlauben dabei auch Unterteilungen auf 1/8 eines Buchstabenformates genau. Liegen die Balken waagerecht, ist es sehr bequem, PRINT zu verwenden, bei senkrechten ist das POKEn in das Video-RAM einfacher; man kann aber beide Methoden in beiden Fällen verwenden. Nehmen wir an, die darzustellende Zahl sei bereits auf den Zahlenbereich bis rund 320 zurechtgestutzt ("normiert"). Die Zahl der ganzen Felder erhält man mit Division durch 8, der Rest gibt an, welches Zeichen das verbleibende Feld gerade zu der richtigen Anzahl von Achteln ausfüllt. Für den ganzzahligen Anteil kann man eine Schleife nehmen, muß sie aber umgehen, wenn dieser Anteil null ist. Geben Sie bitte das Programmbeispiel ein: *

```
10 REM BALKENGRAFIK WAAGERECHT MIT PRINT
20 :
30 DATA "■ ","■ ","■ ","■ ","■ ","■▌","■▏","■▏"
40 FOR I=0 TO 7:READ G$(I):NEXT
50 X=160+150*COS(T)*EXP(-T/10)
60 XG=INT(X/8):XF=X-8*XG
70 IF XG<1 THEN 90
80 FOR I=1 TO XG:PRINT"■ ";:NEXT
90 PRINT G$(XF)
100 T=T+.2:GOTO 50
```

**Da aus der AusLISTung die Tasten nicht eindeutig hervorgehen: Die Reihenfolge ist: Leertaste, G H J K L N M, jeweils mit Commodore-Taste und mit Reverse On bzw. Off.*

In Zeile 6Ø kann man verschiedene Zeichen verwenden: Das reverse Leerzeichen füllt die Balken ganz aus; reverse T-Winkel gestatten dagegen ein Abzählen der ganzzahligen Anteile und eine deutliche Trennung von einem Balken der nächsten Zeile, d.h. sie hinterlassen ein Raster. Einen ähnlichen Effekt hat auch der reverse Vollkreis.

Vergessen Sie aber nicht, auch die Farbe zu POKEn !
Zur Sicherheit ein Beispiel für eine Säule von unten nach oben:



```
10 REM HISTOGRAMM SENKRECHT MIT POKE
20 :
30 DATA 32,100,111,121,98,248,247,227
40 FOR I=0 TO 7:READ G(I):NEXT
50 PRINT"□":FOR X=0 TO 39
60 Y=100+100*COS(X)*EXP(-X/5)
70 YG=INT(Y/8):YF=Y-8*YG
80 IF YG<1 THEN 130
90 FOR I=1 TO YG
100 POKE 2024+X-40*I,160
110 POKE 56296+X-40*I,1
120 NEXT
130 POKE 2024+X-40*(YG+1),G(YF)
140 POKE56296+X-40*(YG+1),1
150 NEXT
160 GOTO 160
```

Falls nacheinander immer wieder neue Säulen an die gleiche Stelle gesetzt werden sollen, braucht die jeweils alte nicht unbedingt ganz gelöscht zu werden (das würde auf störende Weise flackern); es genügt eine Schleife, die den Teil über der neuen Säule löscht.

Ein anderer grafischer Trick mit normalen Tastatur-Grafikzeichen benutzt die verschieden hohen waagerechten oder die unterschiedlich weit nach rechts gerückten senkrechten dünnen Striche. Ihre Bildschirm-Codenummern* sind:

---

*vgl. Anhang Seite 234 ff.

waagerecht:(unten)1ØØ,82,7Ø,64,67,68,69,99(oben), senkrecht:(links) 1Ø1,84,71,66,93,72,89,1Ø3(rechts). Mit ihnen kann man relativ bequem Funktionen darstellen, bei denen waagerecht ein grobes Raster genügt, senkrecht aber ein feines erwünscht ist, oder umgekehrt.

MITTELFEINES

Viertelgrafik 80 x 50

Der Vorteil dieser Grafik ist, daß sie sich mit den auf der Tastatur vorhandenen Zeichen begnügt (also den Zeichensatz nicht neu definieren muß), aber dennoch 4ØØØ Punkte auflöst

Zunächst werden die nötigen Grafikzeichen (es sind 16 an der Zahl, da jedes Viertel eines Zeichenplatzes voll oder leer sein darf, und weil 2^4=16 ist) vernünftig numeriert:

Ø 1 2 3 4 5 6 7 8 9 1Ø 11 12 13 14 15

Das sind natürlich Binärzahlen, bei denen jedes Bit eines der vier Viertel anzeigt, und zwar mit der Übersetzung ins Dezimale:

1	2
4	8

Leider entsprechen die Bildschirmcode-Nummern dem nicht, so daß man eine Zuordnungsfunktion G%() und ihre Umkehrung H%() braucht: G% ordnet jeder der Zahlen Ø bis 15 den Bildschirmcode des zugehörigen Grafikzeichens zu, H% umgekehrt.

Die Koordinaten eines Punktes seien XX und YY (von links unten an gezählt, XX von Ø bis 79, YY von Ø bis 49). Für das gröbere Raster der 4Ø Spalten und 25 Zeilen muß man einfach ohne Rest durch 2 teilen. Die Reste (Ø oder 1) bestimmen nun,

```
2 REM  VIERTEL-GRAFIK
3 :
4 REM VG=0 ABFRAGE
5 REM VG=1 SETZEN
6 REM VG=2 LOESCHEN
7 REM VG=3 INVERTIEREN
8 :
9 REM VORLAUF ZUR VIERTELGRAFIK
10 DIM G%(15),H%(255):P0=1024+960:P1=55296+960
20 DATA 32,126,124,226,123,97,255,236,108,127,225,251,98,252,254,160
30 FOR I=0 TO 15:READ G%(I):H%(G%(I))=I:NEXT
40 :
299 :
300 REM DEMO
301 :
310 PRINT"♥":FOR I=15 TO 25 STEP 2:FOR J=0 TO 24
320 POKE 1024+J*40+I,160:POKE 55296+J*40+I,6:NEXT:NEXT
325 FOR I=26 TO 39:FOR J=0 TO 24
326 POKE 1024+J*40+I,160:POKE 55296+J*40+I,6:NEXT:NEXT
330 FOR XX=0 TO 79
340 Y=8*SIN(XX/5)
345 YY=8+Y:VG=1:GOSUB 63000
350 YY=25+Y:VG=2:GOSUB 63000
360 YY=42+Y:VG=3:GOSUB 63000
370 NEXT
380 GOTO 380
62998 :
62999 REM UNTERPROGRAMM-SCHLEIFE, EINGABEN XX,YY,VG; AUSGABE VT
63000 XI=INT(XX/2):YI=INT(YY/2)
63005 IF XX<0 OR YY<0 OR XX=>80 OR YY=>50 THEN RETURN
63010 G=(1+(XX AND 1))*(4-3*(YY AND 1))
63020 P=XI-INT(YI)*40:G1=PEEK(P+P0):H=H%(G1)
63030 VT=SGN(G AND H)
63040 IF(VG=1)OR(VG=3 AND VT=0)THEN G2=G%(H OR G):POKE P0+P,G2:POKE P1+P,6
63050 IF(VG=2)OR(VG=3 AND VT=1)THEN G2=G%(H-(H AND G)):POKE P0+P,G2:POKE P1+P,6
63060 RETURN
```

welches der vier Viertel des angesprochenen Platzes nun ausgefüllt werden muß. Wir gehen natürlich davon aus, daß in den anderen drei schon etwas sitzt, was auch bleiben muß. Darum wird das alte Zeichen abgefragt (mit PEEK), mit H% in unseren Code umgewandelt und mit OR dazugetan. Das entstehende neue Zeichen, in dem der neue Punkt nun auf jeden Fall enthalten ist (unabhängig davon, ob er es vorher auch schon war), wird nun mit POKE nach Umwandlung mit G%() in den Bildschirm gesetzt, sowie in das Farb-RAM. Das Hauptprogramm muß für jeden einzelnen Punkt die Koordinaten als XX (zwischen Ø und 79) und YY (von Ø bis 49; Nullpunkt links unten) bereitstellen und auch dafür sorgen, daß dieser Bereich nicht überschritten wird; außerdem muß es den Modus VG festlegen:

VG = Ø bedeutet Abfragen des Punktes
(VT ist 1 bzw. Ø nach der Abfrage)
VG = 1 bedeutet Setzen (in Tintenfarbe)
VG = 2 bedeutet Löschen (zur Papierfarbe)
VG = 3 bedeutet Invertieren (Wechseln der Farbe)

Die Zeilen 1Ø bis 3Ø bereiten das Unterprogramm vor und dürfen nur einmal durchlaufen werden. Das Hauptprogramm in dem hier wiedergegebenen Beispiel zeichnet erst helle und dunkle Streifen und dann auf diesen Hintergrund drei Sinuskurven, eine mit Setzen (nur auf leerem Hintergrund sichtbar), eine mit Löschen und eine mit Invertieren. So verlockend das Invertieren ist: Man muß gewaltig aufpassen, daß eine Kurve aus zu engen Punkten sich nicht teilweise selbst invertiert; das sieht dann sehr löcherig aus (Gegenrezept: Schrittweite groß genug wählen oder noch besser: von der Steigung abhängig machen !).

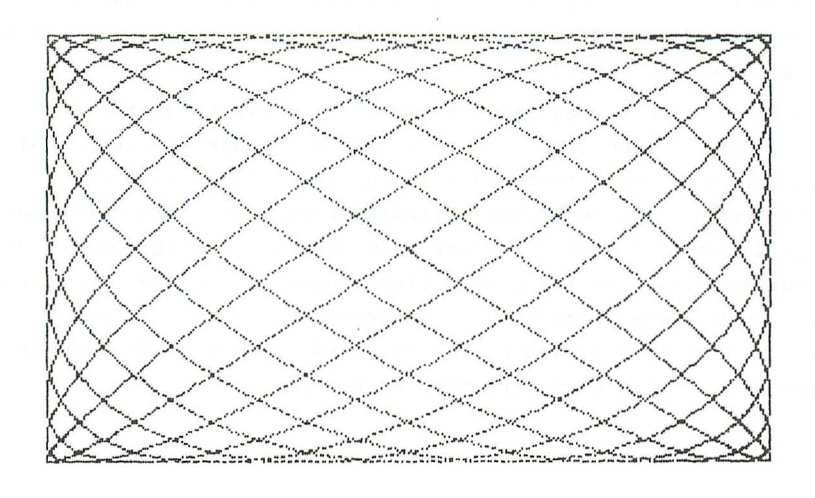
Falls Ihnen Zeile 63Ø1Ø seltsam vorkommt: Das VerUNDen mit 1 ist eine Abfrage auf ungerade Zahlen; es liefert Ø für gerade und 1 für ungerade XX bzw. YY. Allgemein gibt A AND 2↑N-1 den Rest, der sich beim Teilen von A durch 2↑N ergibt.

PUNKTGRAFIK OHNE SIMON

Damit niemand sagen kann, der C 64 könne keine Hochauflösung bringen, ohne daß sein BASIC mit SIMON oder anderen Erweiterungen ergänzt wird, soll das "Bit-Mapping" hier kurz vorgeführt werden. Andererseits sollte das kein Grund sein, auf SIMON's BASIC zu verzichten. Die Erklärung wird daher hier nur sehr knapp sein. Das Beispiel zeigt eine Lissajous-Figur, also eine zweidimensionale Überlagerung von Schwingungen mit (etwas) verschiedenen Frequenzen (Zeilen 3Ø bis 4Ø als Endlosschleife). Die anderen Zeilen bilden ein Vor- und ein Unterprogramm für die nötigen Umschaltungen und das Zeichnen einzelner Punkte:
1Ø-2Ø Einschalten der Feingrafik und Wahl des Speicherbereichs,
25-26 Löschen des Speicherbereichs für die 64000 Punkte und
  Setzen von 1ØØØ Kombinationen aus Vorder- und Hintergrundfarbe,
63ØØØ-63Ø1Ø Aufspalten der Koordinaten in Grob- und Feinanteile,
63Ø2Ø P ist die Nummer des betroffenen Bytes im Bit-Mapping-
  Speicher (für je 8 waagerecht benachbarte Punkte zuständig),
  PE der bisherige Inhalt
63Ø3Ø PN ist ein Byte, das nur den neuen Punkte enthält, es
  muß also mit dem bisherigen Inhalt verODERt werden und an die
  alte Stelle P gePOKEt werden.

Wenn Sie dieses Programm mit dem für die Viertelgrafik vergleichen, können Sie es auch so erweitern, daß es Punkte abfragt, löscht oder invertiert. Da letzten Endes alle Linien oder Flächen aus Rasterpunkten bestehen, können Sie auch Schleifen für Geraden, Kreise, Rechtecke usw. machen. Die Ausführung dieser Figuren dauert in BASIC natürlich länger als in den Erweiterungen, in denen diese Prozeduren als Maschinenroutinen enthalten sind; es sind aber alle Bilder, die der C 64 auf dem Bildschirm erzeugen kann, mit diesem Verfahren machbar.

```
5 REM PUNKTGRAFIK OHNE SIMON
6 :
10 EI=53265:POKE EI,PEEK(EI) OR 32
20 BA=8192:POKE53272,PEEK(53272) OR 8
25 FOR I=0 TO 2↑13:POKE BA+I,0:NEXT
26 FOR I=0 TO 999: POKE 1024+I,16:NEXT
27 :
28 REM  LISSAJOUS-FIGUR ALS DEMO
29 :
30 X=160+159*SIN(W):Y=100+99*SIN(W*1.1):GOSUB 63000
40 W=W+.01:GOTO 30
62997 :
62998 REM PUNKTGRAFIK
62999 :
63000 XG=INT(X/8):XF=INT(X-8*XG)
63010 YG=INT(Y/8):YF=INT(Y-8*YG)
63020 P=BA+YG*320+YF+XG*8:PE=PEEK(P)
63030 PN=2↑(7-XF):POKE P,PE OR PN
63040 RETURN
```



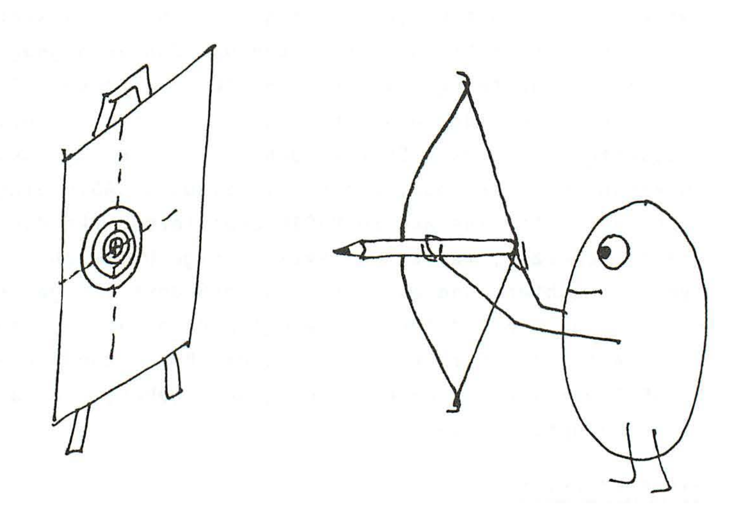
SIMON MACHT'S BEQUEM
Feingrafik mit HIRES

Mit dem eingebauten Microsoft-BASIC ist der Commodore 64 wie ein Luxus-Auto mit dem Armaturenbrett einer "Ente": Man kann zwar alle wichtigen Dinge tun, aber es ist teilweise sehr unbequem: mit POKE kann man zwar jeden Punkt setzen, aber es ist bei Linien oder Ellipsen in BASIC sehr langsam und außerdem umständlich (und für Unterrichtszwecke zu sehr mit maschinentypischen Dingen belastet). Wenn man sich nicht nur mit Texten und Zahlen begnügen will, ist daher eine feingrafikfähige BASIC-Erweiterung als Zubehör noch notwendiger als ein Programmspeicher (Floppy oder Datasette). In diesem Buch werden einige wichtige Grafikanweisungen besprochen, wie sie in SIMON's BASIC eingebaut sind. Wenn Sie eine andere BASIC-Erweiterung für die Feingrafik benutzen, müssen Sie evtl. einige Unterschiede der Syntax beachten, und es kann sein, daß dort einige Tricks nicht gehen, aber dafür vielleicht andere, die es hier nicht gibt. Auf jeden Fall können Ihnen die Bemerkungen über die SIMON-Anweisungen auch dann Anregungen geben, und das ist ja das Hauptziel dieses Buches.

Das Bildformat

Der Bildschirm des C 64 ist in 320 x 200 Punkte aufgeteilt, die auch im Textmodus benutzt werden (jeder Buchstabe belegt dabei 8 x 8 Punkte). Im Multicolor-Modus werden je zwei waagerecht benachbarte Punkte zusammengefaßt: 160 x 200 Punkte. Die Koordinaten (also die Zahlen, die die Positionen der Punkte beschreiben) werden von links nach rechts (0 bis 319 bzw. 0 bis 159) und von oben nach unten (0 bis 199) numeriert.Wenn Sie also ein Programm in den Multicolor-Modus oder zurück umändern wollen, müssen Sie alle waagerechten Koordinaten um den Faktor 2 ändern, am besten direkt in den Grafikanweisungen und nicht in den Rechenanweisungen. Beachten Sie bitte auch, daß die senkrechten Koordinaten in der Mathematik von unten nach oben gezählt werden; statt Y müssen Sie also 199-Y nehmen, um das auszugleichen.

Auf einem richtig eingestellten Fernsehgerät erscheint die Höhe um etwa 15 % zu groß, verglichen mit der Breite, wenn Sie gleiche Zahlen nehmen. Beim Drucken mit dem VC 1525 oder VC 1515 ist das Verhältnis fast genau Eins. Sollen Ihre Bilder hauptsächlich auf dem Bildschirm genau aussehen, multiplizieren Sie alle senkrechten Koordinaten mit 0.85, um das auszugleichen. Die Programmbeispiele in diesem Buch sind teilweise auf die Wiedergabe über den Drucker zugeschnitten.



AUF DEN PUNKT GEBRACHT
PLOT

PLOT x,y,z

Die PLOT-Anweisung zeichnet einen einzelnen Punkt mit den Koordinaten x und y (die kleinen Buchstaben stehen hier stellvertretend für irgendwelche Zahlen oder Variable, die in Ihrem Programm diese Bedeutung haben !). z ist dabei der Zeichentyp; damit ist gemeint: die indirekte Farbwahl bzw. die Option der Umkehrung (Inversion). Die Hintergrundfarbe (die als zweites Argument in HIRES bestimmt

wird oder in den 1000 Speicherplätzen ab 48*1024 durch die vier niedrigen Bits für jedes der 1000 Buchstabenfelder einzeln bestimmt werden kann) hat die Zeichentyp-Nr. 0. 1 ist entsprechend die (erste bzw. einzige) Vordergrundfarbe (obere Bits in den genannten Speicherplätzen oder erstes Argument in der HIRES-Anweisung). 2 und 3 sind nun im Multicolor-Modus die weiteren Farben, die dort in der MULTI-Anweisung und in LOW COL gewählt werden können.Das Setzen der Hintergrundfarbe (0) bedeutet im allgemeinen ein Löschen eines Zeichens oder eine Negativzeichnung auf einer bereits ausgefüllten Fläche. Beachten Sie bitte, daß Sie alle 16 Farben für Vorder- und Hintergrund nehmen können, daß also Weiß nicht notwendig der Hintergrund sein muß.

UMGEKEHRT IST NICHT VERKEHRT
Inversion mit Zeichentyp 2

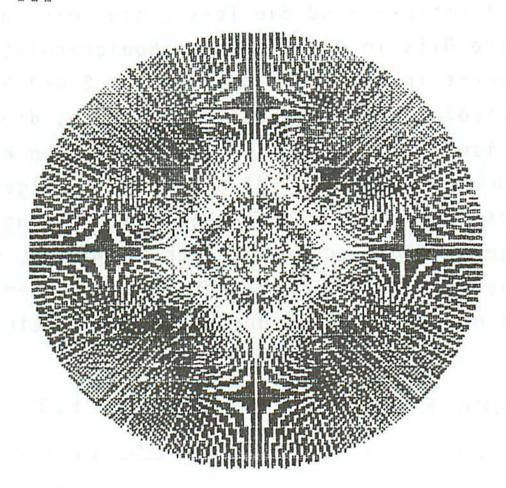
Dieser Zeichentyp (Code 4 bei Multicolor, sonst 2) setzt oder löscht einen Punkt stets entgegengesetzt zu seiner bisherigen Besetzung, auch innerhalb einer Linie oder sogar eines "Blocks" für jeden Punkt einzeln. Das ist sehr praktisch, gibt aber spezielle Effekte, die meist unerwünscht sind, wenn der gleiche Punkt dabei in nicht absehbarer Weise mehrfach erwischt wird. Wenn dabei eine versteckte Regelmäßigkeit auftritt, kann man das auch zu einem grafischen Trick nutzen: das folgende Bild sieht sehr raffiniert aus, besteht aber nur aus vielen einfachen Linien mit Invertierung.

Bei Multicolor wird 0 zu 3 invertiert, 1 zu 2 und jeweils umgekehrt. Der Grund liegt einfach darin, daß die beiden Bits, in denen die (indirekten) Farbcodes 0 bis 3 vorliegen (00, 01, 10, 11) einzeln invertiert werden, also:

00 = 0 wird zu 11 = 3 und umgekehrt,
01 = 1 wird zu 10 = 2 und umgekehrt

(die zweistelligen Zahlen sind binär gemeint, die einstelligen dezimal).

```
100 HIRES15,0
110 FORI=0TO2*πSTEP.015:X=100*COS(I):Y=100*SIN(I)
120 LINE160,100,160+X,100-Y,2:NEXT
130 GOTO 130
```

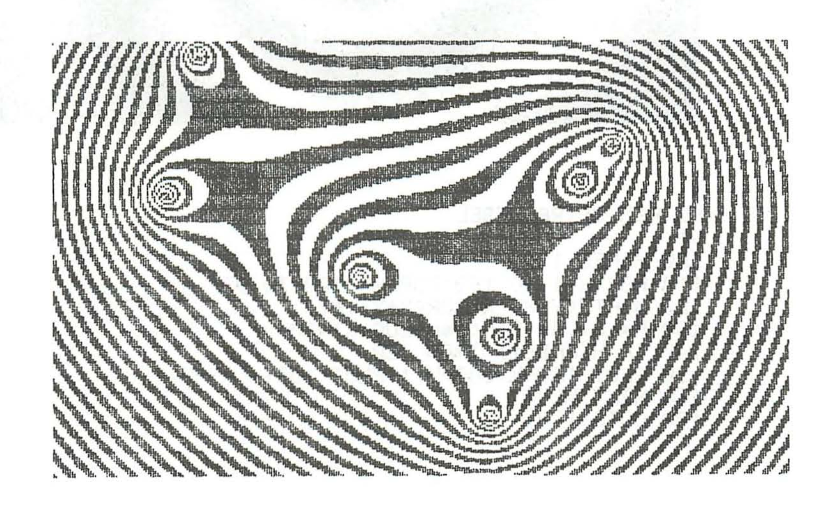


PUNKT FUR PUNKT

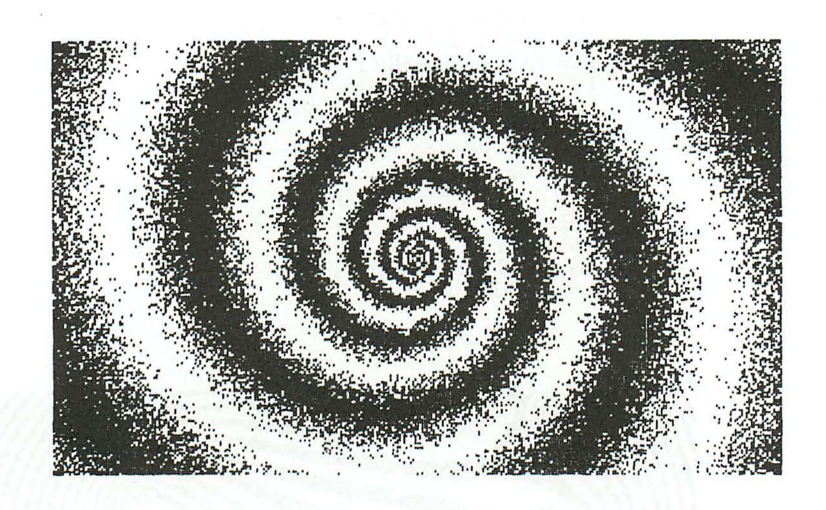
Bilder mit PLOT

Obwohl es Anweisungen für kompliziertere Dinge als einzelne Punkte gibt, z.B. Linien, Ellipsen usw., kann man besonders komplizierte (vor allem flächenfüllende) Bilder nur mit PLOT machen. Das folgende Bild zeigt 7 zufällige Punkte. Die Grenzen zwischen Hell und Dunkel sind Linien, auf denen jeweils das Produkt der Entfernungen zu allen diesen sieben Punkten gleich ist. Im Programm geht das so vor sich, daß zu jedem der 320 x 200 Punkte erst einmal diese Entfernungen (mit Satz von Pythagoras) und deren Produkt ausgerechnet werden. Ist nun dieses Produkt (das noch durch 3 geteilt wird) eine ungerade Zahl ("AND 1" angelt genau diese heraus), wird der Punkt gesetzt. Im Falle von nur 2 Punkten nennt man die Kurven zwischen den Flächen Cassini-Kurven und deren Spezialfall, der wie eine 8 aussieht, die Lemniskate.

```
100 HIRES 1,0:N=7
110 FOR I=1 TO N
120 X(I)=320*RND(1)
130 Y(I)=200*RND(1)
140 CIRCLE X(I),Y(I),2,2,1
150 NEXT
160 FOR X=0 TO 319
170 FOR Y=0 TO 199
180 R=1:FOR I=1 TO N
190 R=R*((X-X(I))↑2+(Y-Y(I))↑2)
200 NEXT:R=R↑(1/N/2)/3
210 PLOT X,Y,R AND 1:NEXT:NEXT
220 GOTO 220
```



PLOT kann auch zum "stufenlosen" Übergang zwischen Hell und Dunkel verwendet werden; natürlich gilt das dann nicht für die einzelnen Punkte, sondern nur für Bereiche, in denen zufällig Punkte gesetzt werden, und zwar um so dichter, je heller oder dunkler es sein soll. Das folgende kleine Programm zeichnet mit dem Zufallsgenerator einen Spiralnebel. Dabei bilden die ganz hellen und die ganz dunklen Kurven logarithmische Spiralen, und die ganze Figur (abgesehen von den Details der Rasterung und der zufälligen Punkte) geht bei Drehung und gleichzeitiger Maßstabsänderung in sich selbst über.



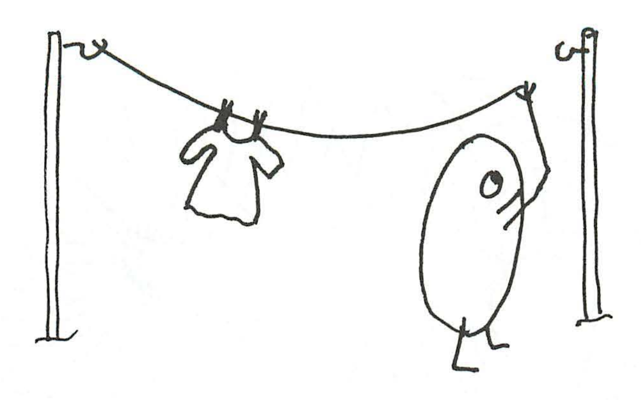
```
90 REM SPIRALNEBEL
91 :
100 HIRES 1,0
110 FOR X=1 TO 160:FOR Y=-100 TO 99
120 W=ATN(Y/X):R=SQR(X*X+Y*Y)
130 H=.5+.5*(SIN(2*W+LOG(R)*10))
140 IF H<RND(1) THEN 160
150 PLOT 160-X,100+Y,1:PLOT 159+X,99-Y,1
160 NEXT:NEXT
170 GOTO 170
```

VON EINEM ORT ZUM ANDERN

Geraden mit LINE

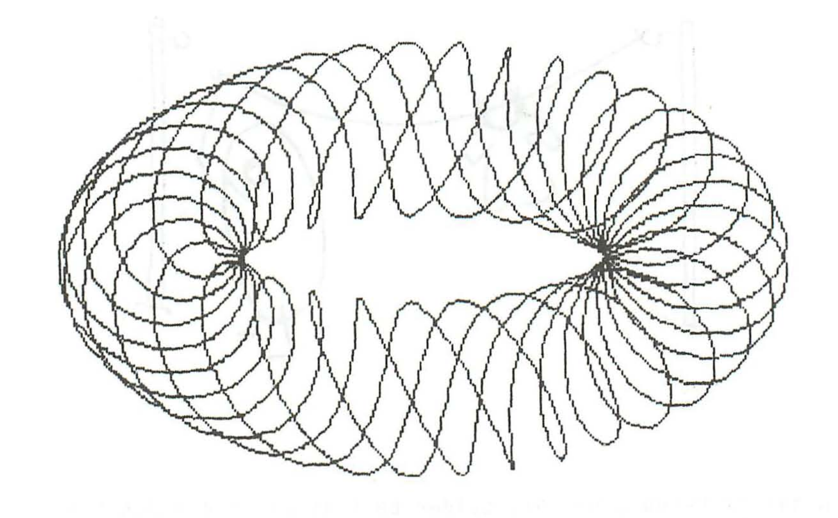
LINE x1,y1,x2,y2,z

Gerade Linien werden durch ihre beiden Endpunkte bestimmt. Beide sollten im zulässigen Bildschirmbereich liegen, weil sonst bei zu großen Koordinaten der entsprechende Endpunkt verlegt wird und bei negativen das Programm mit ILLEGAL



QUANTITY ERROR abbricht. Leider besteht nicht die Möglichkeit, mit nur einem Koordinatenpaar eine Verbindung zum letzten gezeichneten Punkt herzustellen, wie sie z.B. beim DRAW des VC-20-Super-Expanders sehr bequem ist. Allerdings ist in beiden Fällen das Hauptproblem, beim Start eines Linienzuges vorher den ersten Punkt anderweitig richtig zu besetzen. Beachten Sie im Programm TORUS-WENDEL die Zeile 15Ø: der Punkt X,Y wird mit dem "alten" Punkt XA,YA verbunden und anschließend "umbenannt", damit er im nächsten Schleifendurchlauf als "alter" zur Verfügung steht. Vor der Schleife müssen XA und YA (falls sie nicht Ø sein sollen) für den ersten Punkt korrekt vorgegeben werden. Das Bild zeigt eine Linie, die sich viermal um einen Rettungsring windet, gesehen in Parallelperspektive (Schrägbild).

```
100 REM TORUS-WENDEL
101 :
105 HIRES 1,0
110 XA=319:YA=100
120 FOR W=0 TO 8*π STEP .01
130 X=160+120*COS(W)+40*COS(W*8.25)*COS(W)
140 Y=100-60*SIN(W)-40*SIN(W*8.25)
150 LINE X,Y,XA,YA,1:XA=X:YA=Y
160 NEXT
170 GOTO 170
```



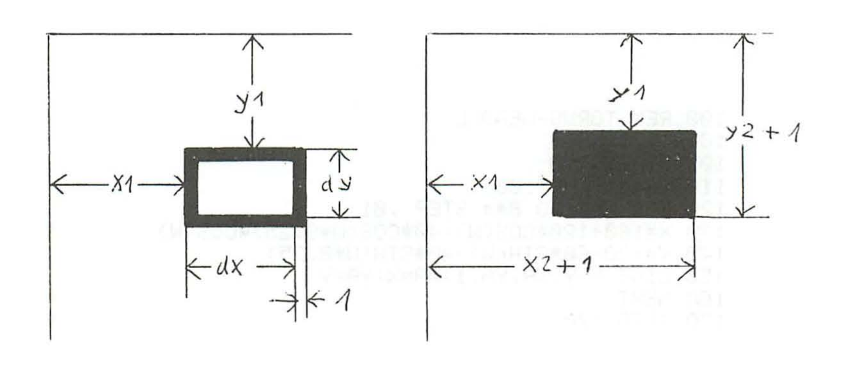
ALLES WAS RECHT(-ECKIG) IST

BLOCK und REC

Leere und volle Rechtecke mit waagerechten und senkrechten Seiten können jeweils mit einer einzigen Anweisung erzeugt werden; in beiden Fällen benötigt man die linke obere Ecke; dann allerdings wird es inkonsequent: Das volle Rechteck

```
BLOCK x1,y1,x2,y2,z
```

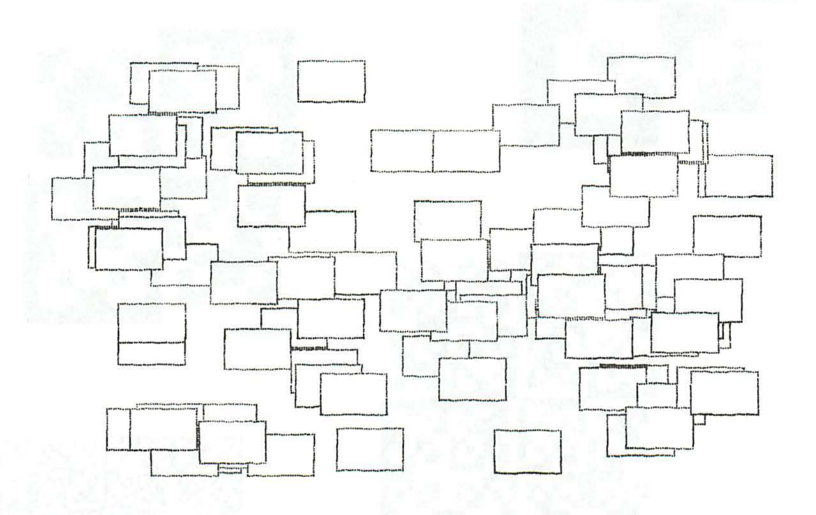
verlangt als zweite Angabe die rechte untere Ecke. Liegt



sie nicht unter der anderen oder nicht rechts von ihr, so bleibt von dem Rechteck nur eine Gerade oder sogar nur ein Punkt. Das leere Rechteck (Rahmen)

REC x1,y1,dx,dy,z

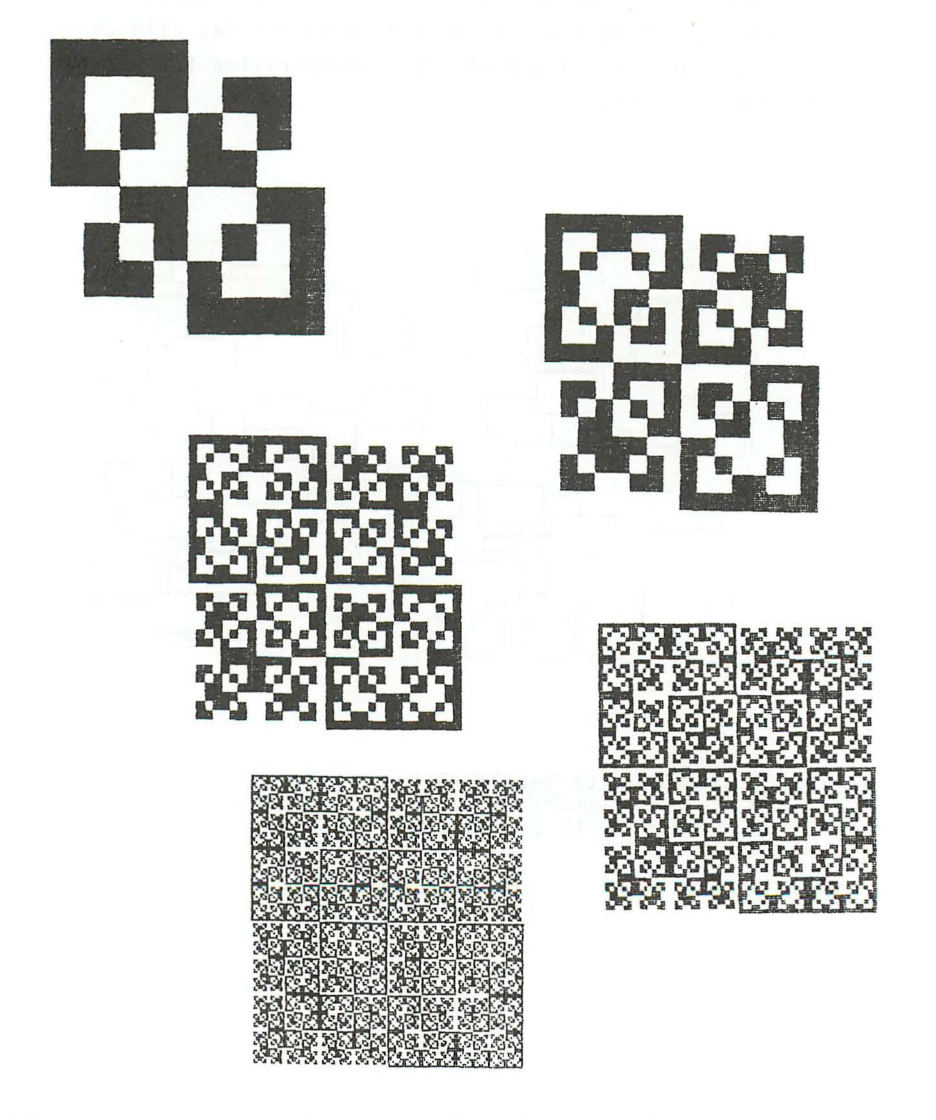
benötigt dagegen als zweites Zahlenpaar die Breite und Höhe. Jede Festlegung für sich ist sinnvoll, dieser "Kompromiß" scheint mir jedoch nicht besonders günstig für das Gedächtnis zu sein. Das kleine Programm würfelt sich Positionen für Rechtecke und zeichnet sie so, daß sie die "darunterliegenden" gegebenenfalls verdecken. Wenn man das Bild betrachtet, könnte man das Programm möglicherweise für wesentlich länger halten.

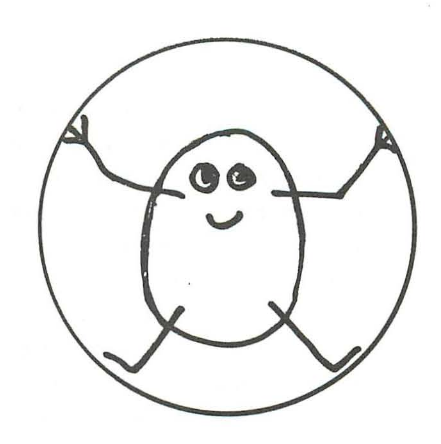


```
100 HIRES 1,0
110 X=290*RND(1):Y=180*RND(1)
120 BLOCK X,Y,X+29,Y+19,0
130 REC X,Y,29,19,1:GOTO 110
```

Inversionen

Diese Computergrafik können Sie leicht nachmachen. Sie brauchen dazu außer HIRES, der Wertzuweisung = und einigen Grundrechnungsarten nur die Anweisung BLOCK mit dem Zeichentyp 2 (Invertierung) und einige Schleifen mit FOR TO NEXT. Es werden jeweils ganze Quadrate invertiert, und das ganze Bild besteht aus 128 mal 128 Punkten.

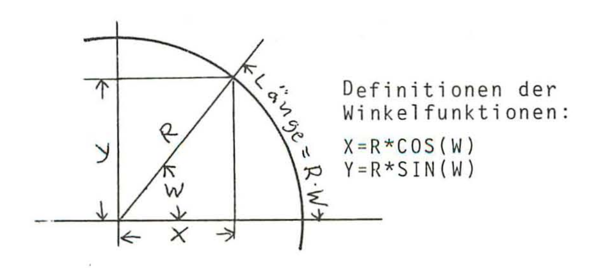




RUNDE SACHEN
ARC und CIRCLE

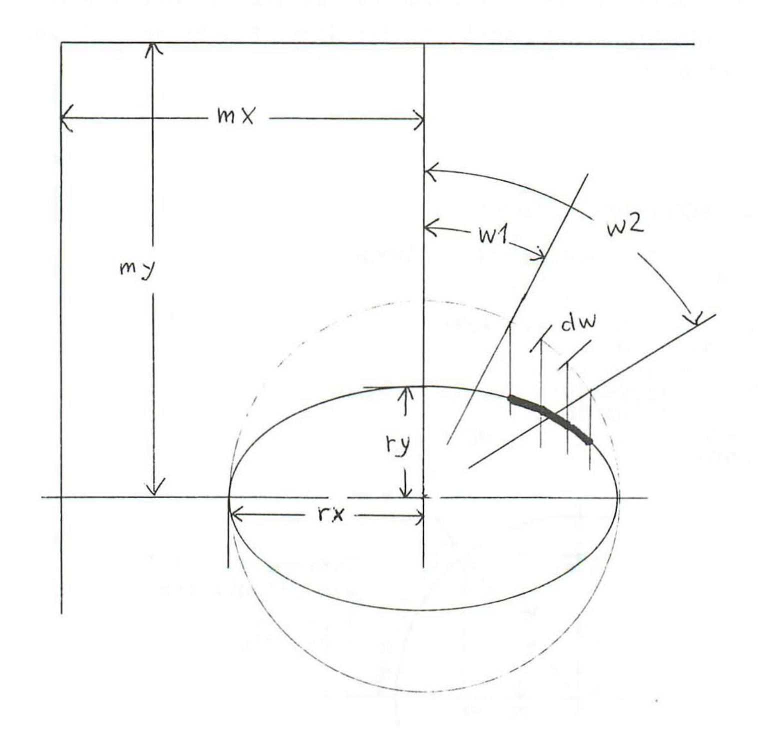
Um einen Kreis zu zeichnen, nimmt man normalerweise einen Zirkel. Soll man aber einzelne Punkte auf ihm mit Koordinaten angeben, so braucht man die Winkelfunktionen SIN und COS, etwa so:

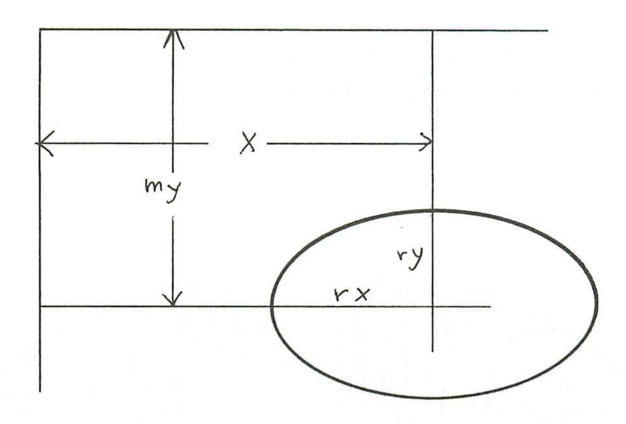
```
90 REM REGELMAESSIGES VIELECK
91 :
100 INPUT"ZAHL DER ECKEN   17■■■■■";N
110 HIRES 1,0
120 FOR I=0 TO N-1
125 W1=I*2*π/N:W2=(I+1)*2*π/N
130 X1=160+100*COS(W1)
135 X2=160+100*COS(W2)
140 Y1=100+100*SIN(W1)
145 Y2=100+100*SIN(W2)
150 LINE X1,Y1,X2,Y2,1:NEXT
160 GOTO 160
```



Das gibt N Punkte, gleichmäßig auf einen Kreisumfang verteilt, und, wenn man sie geradlinig miteinander verbindet, ein regelmäßiges N-Eck. Ist die Eckenzahl N genügend groß, kann man das von einem Kreis nicht unterschieden. Genauer: der Unterschied zwischen N-Eck und richtigem Kreis soll kleiner sein als die Größe der Rasterpunkte. Das hängt durchaus vom Radius ab: Für den vierfachen Radius genügt die doppelte Eckenzahl, und für einen Radius von 100 Einheiten eignet sich ein Vieleck mit 30 bis 40 Ecken. Obwohl wir elegante Anweisungen für Kreise haben, muß man manchmal Kreisbögen mit LINE nach dem obigen Rezept zeichnen, nämlich wenn der Mittelpunkt außerhalb des Bildes liegt.

Ändert man die Breite oder die Höhe des Bildes, entsteht statt des Kreises eine Ellipse mit einer waagerechten und einer senkrechten Symmetrieachse. Auch die Anweisungen ARC und CIRCLE zeichnen Kreise und Kreisbögen als Spezialfälle von Ellipsen(-bögen).



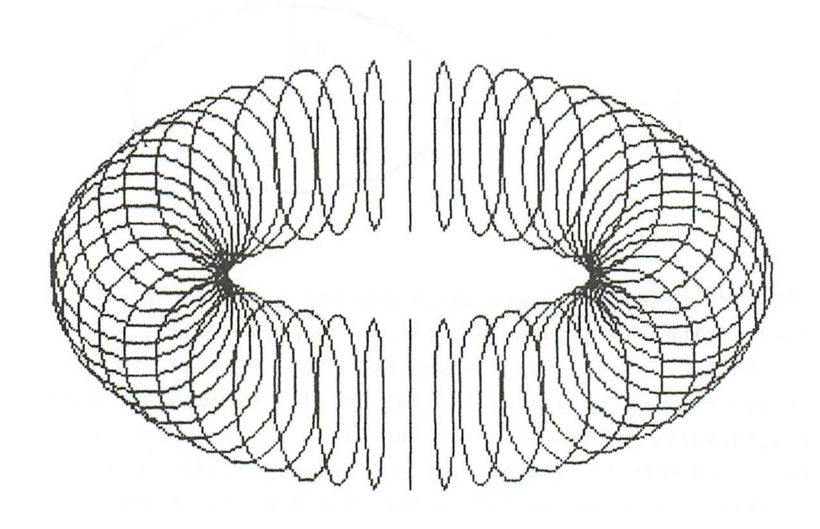


ARC ist die allgemeinere Anweisung und hat daher eine ganze Reihe von Argumenten: ARC mx,my,w1,w2,dw,rx,ry,z mx und my sind die Koordinaten des Mittelpunktes, rx und ry die Halbachsenlängen; für das auf dem VC 1525 mit COPY ausgedruckte Bild müssen sie für einen Kreis gleich sein, für einen korrekt eingestellten Bildschirm aber im Verhältnis 1,15:1 zueinander stehen, damit es einen Kreis gibt. z ist der Zeichentyp, also die indirekte Farbenwahl. Diese Argumente treten auch bei der Anweisung CIRCLE mx,my,rx,ry,z auf. Zusätzlich hat ARC aber noch den Anfangswinkel w1, den Endwinkel w2 und die Winkelschrittweite dw, jeweils in Grad. Der Nullpunkt für w1 und w2 liegt oben, und es geht im Uhrzeigersinn weiter. Mit dw kann man die Genauigkeit der Ellipse, also die Eckenzahl des tatsächlich gezeichneten Vielecks beeinflussen, die sich natürlich auch auf die Zeichengeschwindigkeit auswirkt. Zu empfehlen sind:

Radius	200	100	50	25	12	Einheiten
dw	7	10	15	20	30	Grad oder weniger.

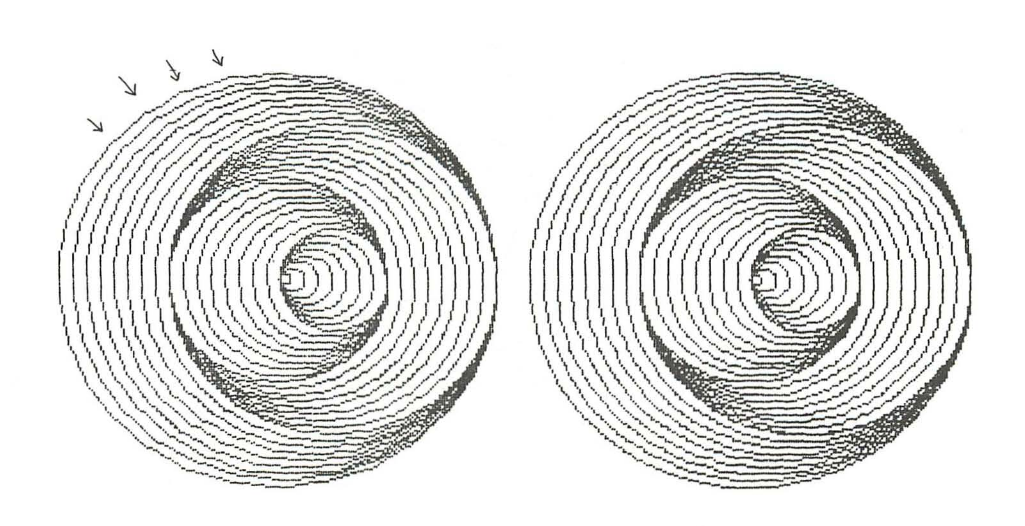
Bei dieser Zuordnung ist die Abweichung zwischen Kreis und regelmäßigem Vieleck in der Größenordnung der Rasterpunkte. Die Anweisung CIRCLE zeichnet nicht nur stets volle Ellipsen, sondern nähert sie durch 32-Ecke an, was bei großen Radien evtl. stört.

Das folgende Bild zeigt Ellipsen, die als perspektivische Darstellung eines Torus (Rettungsringes) gesehen werden können:



```
90 REM TORUS MIT ARC-ANWEISUNG
91 :
100 HIRES 1,0
110 FOR I=0 TO 48
115 W=I*π/24
120 X=160+120*COS(W)
130 Y=100-60*SIN(W)
140 A=ABS(40*COS(W))
150 ARC X,Y,0,360,10,A,40,1
160 NEXT
170 GOTO 170
```

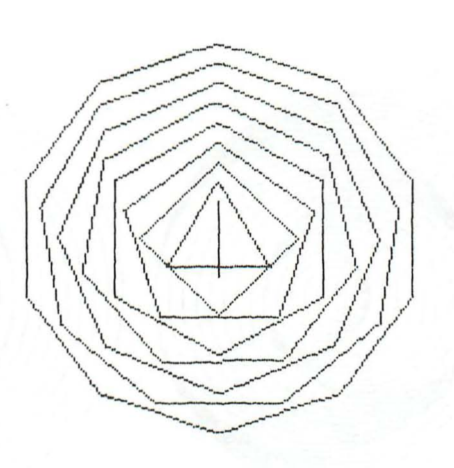
Vergleichen Sie bitte die beiden "gleichen" Bilder: das linke ist mit CIRCLE, das rechte mit ARC erzeugt. Es ist im übrigen ein schönes Beispiel für ein Umklappbild, bei dem unser Gehirn schlagartig zwischen den beiden möglichen räumlichen Deutungen wechselt: denken Sie sich den Mittelpunkt abwechselnd als Turmspitze und als Krater !



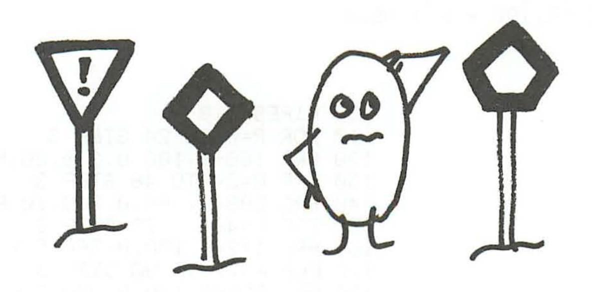
```
100 HIRES 1,0
110 FOR R=0 TO 24 STEP 3
120 CIRCLE 160+R,100,R,R,1:NEXT
130 FOR R=24 TO 48 STEP 3
140 CIRCLE 208-R,100,R,R,1:NEXT
150 FOR R=48 TO 72 STEP 3
160 CIRCLE 112+R,100,R,R,1:NEXT
170 FOR R=72 TO 96 STEP 3
180 CIRCLE 256-R,100,R,R,1:NEXT
200 GOTO 200
```

```
100 HIRES 1,0
110 FOR R=0 TO 24 STEP 3
120 ARC 160+R,100,0,360,20,R,R,1:NEXT
130 FOR R=24 TO 48 STEP 3
140 ARC 208-R,100,0,360,10,R,R,1:NEXT
150 FOR R=48 TO 72 STEP 3
160 ARC 112+R,100,0,360,5,R,R,1:NEXT
170 FOR R=72 TO 96 STEP 3
180 ARC 256-R,100,0,360,3,R,R,1:NEXT
200 GOTO 200
```

Wie Sie hier sehen, kann man mit ARC auch regelmäßige (oder durch Dehnung daraus hervorgehende) Vielecke zeichnen; allerdings setzt das voraus, daß 360 durch die Zahl der Ecken teilbar ist (darum geht dieses Programm auch nur bis 10).



```
100 HIRES 1,0
110 FOR I=2 TO 10
120 ARC 160,100,0,360,360/I,I*9,I*9,1
130 NEXT
200 GOTO 200
```

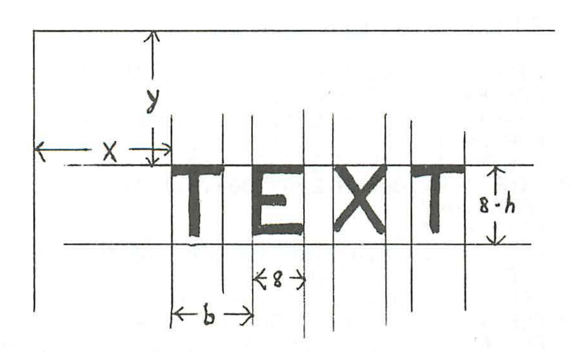


MIXTUM COMPOSITUM

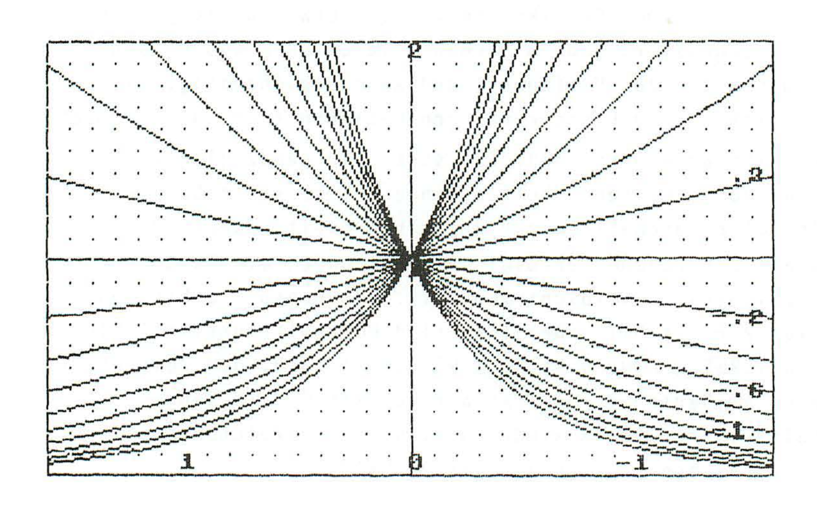
Text in der Feingrafik

Es soll Computer geben, bei denen es Schwierigkeiten macht, eine Grafik mit Text zu versehen. SIMON macht es uns mit zwei Anweisungen sehr bequem und erlaubt einige Freiheiten: Man kann die Buchstaben an jede beliebige Stelle setzen, und zwar im feinen Grafikraster, nicht etwa nur im groben Buchstabenraster, und es stehen alle Zeichen aus beiden Zeichensätzen zur Verfügung, und zwar gleichzeitig, also 512 Zeichen. Die Einschränkungen beziehen sich auf die Größe der Buchstaben: ihre Höhe ist stets ein ganzzahlig-Vielfaches von 8 Rastereinheiten, ihre Breite stets 8, was in Multicolor automatisch zur doppelten Breite führt. Die Auswahl der Zeichen erfolgt bei CHAR durch die Nummer des Zeichens im Bildschirmcode (1. Zeichensatz Ø bis 255, 2. von 256 bis 511). Bei TEXT sind Umschaltfunktion (SHIFT) und REVERSE genau so wirksam wie sonst auch; der Zeichensatz 2 wird am Beginn eines Text-Strings mit CTRL-B eingeschaltet, mit CTRL-A wieder aus, d.h. zum Zeichensatz 1. Die Syntax ergibt sich aus diesem Bild:

CHAR x,y,n,z,h,b   TEXT x,y,t$,z,h,b



In der folgenden Darstellung von Exponentialfunktionen sind einige Zahlen mit TEXT eingefügt (deren Positionen berechnet werden), die Rasterpunkte sind zur besseren Orientierung und zum Ablesen von Funktionswerten durch Invertierung (Zeichentyp 2) eingetragen.




```
90 REM EXPONENTIALFUNKTIONEN
91 :
100 HIRES 1,0:REC 0,0,319,199,1:LINE 160,0,160,199,1
110 FOR E=-2 TO 2 STEP .2
120 XA=-1.6:YA=EXP(-E*1.6):IF YA>2 THEN YA=2
130 FOR X=-1.6 TO 1.6 STEP .01
140 Y=EXP(E*X):IF Y>2 THEN Y=2
150 LINE 160+100*XA,200-100*YA,160+100*X,200-100*Y,1
160 XA=X:YA=Y
170 NEXT:NEXT
180 FOR X=0 TO 310 STEP 10:FOR Y=0 TO 190 STEP 10
190 PLOT X,Y,2:NEXT:NEXT
200 FOR X=-1 TO 1:TEXT 150-100*X,190,STR$(X),1,1,8:NEXT
210 FOR Y=1 TO 2:TEXT 150,201-100*Y,STR$(Y),1,1,8:NEXT
220 FOR E=-1 TO .2 STEP .4:TEXT 292,196-100*EXP(E*1.6),
STR$(E),1,1,8:NEXT
300 GOTO 300
```

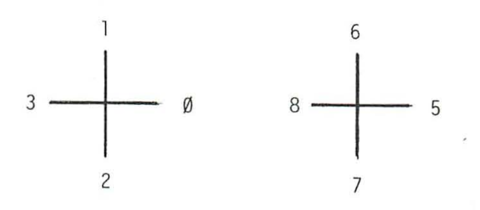
Soll an die gleiche Stelle einer Grafik nacheinander ein unterschiedlicher Text geschrieben werden, so muß die Stelle zwischendurch gelöscht werden, entweder mit dem alten Text oder mit BLOCK.

FEINE BILDCHEN

Bildelemente mit DRAW und ROT

Dieses Anweisungspaar erlaubt, feste Figuren in eine Feingrafik einzubauen. Der Unterschied zu Sprites besteht zum einen in der Art der Festlegung der Form: Man baut sie aus einem Linienzug, der auch teilweise unsichtbar sein kann, auf. Zum anderen bleiben die DRAW-Figuren an der Stelle, an die man sie mit Koordinaten gesetzt hat; die gleiche Figur kann also beliebig oft gleichzeitig im Bild erscheinen (während ein Sprite nur an einer Stelle gleichzeitig ist, allenfalls können alle 8 Sprites die gleiche Figur haben).

Die DRAW-Figur wird in einem String (oder durch Verkettung mehrerer Strings mit + festgelegt, dabei bedeuten die Zahlen: unsichtbar sichtbar

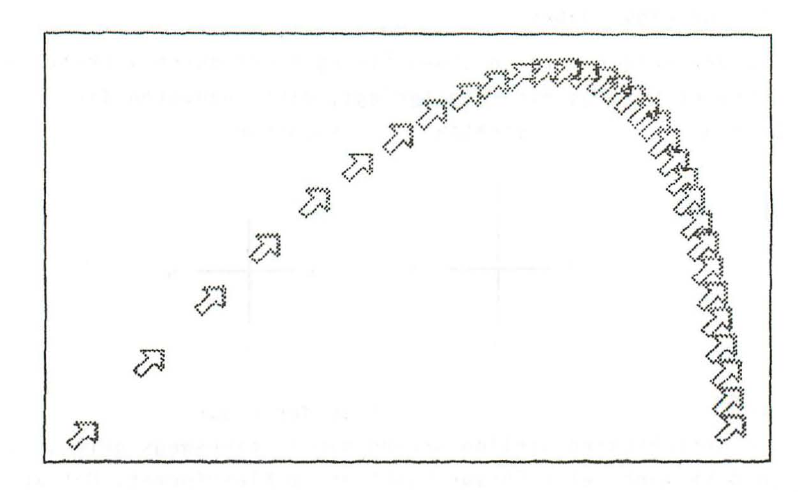


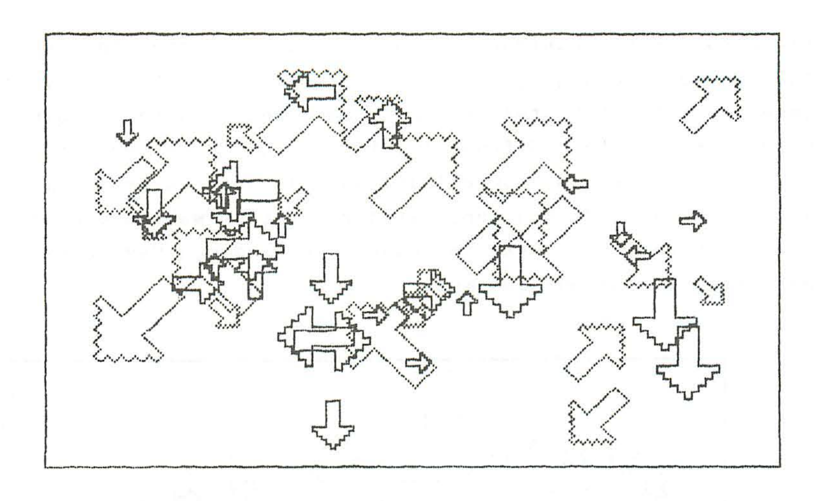
9 Ende der Figur

Die unsichtbaren Stellen werden dabei keineswegs gelöscht, sondern haben eine Cursor-Funktion im Kleinformat. Mit ROT

werden die Richtung und die Größe bestimmt: Der Winkel w zählt im Uhrzeigersinne in Schritten von je 45° von oben angefangen, also von Ø bis 7, die Größe von 1 bis 255 (Ø gibt Unsinn !). Die diagonalen Richtungen ändern die Größe automatisch um den Faktor 1,4 (nämlich $\sqrt{2}$). Die folgenden beiden Beispiele zeigen mit der gleichen Figur (nämlich einem Pfeil) ein Beispiel aus der Physik: schräger Wurf mit (geschwindigkeitsproportionaler) Reibung, wobei andererseits davon abgesehen wird, daß der "Fahrtwind" auch die Drehrichtung des Pfeiles beeinflussen müßte - und ein kleines Zufallsbild mit dem beziehungsreichen Titel "Wohin ?".

```
90 REM WURF MIT REIBUNG
91 :
100 HIRES 1,0:X=10:Y=10:A=-1.5:VX=33:VY=39
105 REC 0,0,319,199,1
110 A$="555666666555686868686878787878755577777779"
120 IF Y<180 THEN ROT 1,1:DRAW A$,X,200-Y,1
130 VY=VY*.9+A:VX=VX*.9:Y=Y+VY:X=X+VX
140 IF X<320 AND Y>0 THEN 120
200 GOTO 200
```

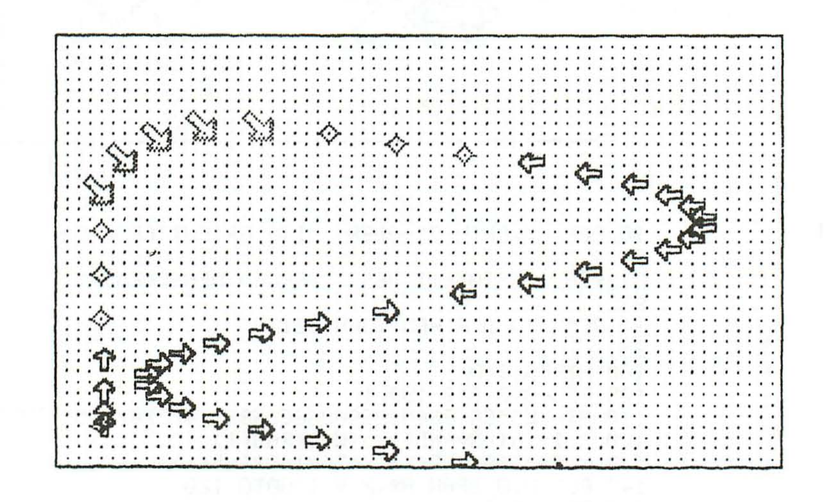




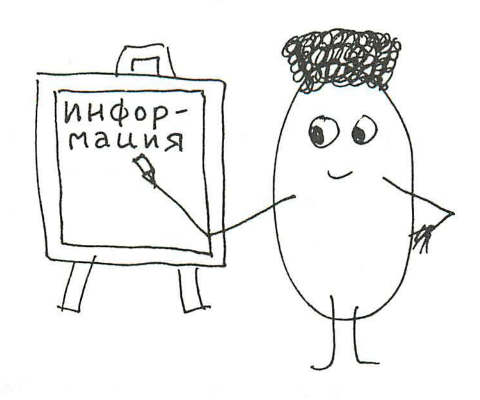
```
90 REM  "WOHIN ?" ODER: "DIE ORIENTIERUNGS-
91 :
92 REM  LOSIGKEIT DES MODERNEN MENSCHEN
93 :
94 REM  IN DER HEUTIGEN ZEIT"
95 :
100 HIRES 1,0
105 REC 0,0,319,199,1
110 A$="5556666665556868686868787878787555777779"
120 G=1+INT(3*RND(1)):W=8*RND(1)
130 X=30+260*RND(1):Y=30+140*RND(1)
140 ROT W,G:DRAW A$,X,Y,1:GOTO 120
200 GOTO 200
```

Schließlich noch eine spielerische physikalische Anwendung: Das Beschleunigungsspiel mit den Joysticks läßt unseren Pfeil in einem feinen Raster wandern, und zwar bestimmt der Joystick nicht etwa die Geschwindigkeit, sondern die Änderung der Geschwindigkeit, also die Beschleunigung. An den Pfeilen können Sie im Bild diese Richtung auch später noch erkennen. Zeigt ein Pfeil nach rechts, so ist der folgende Schritt um eine Einheit länger nach rechts (oder kürzer nach links), ebenso für die anderen Richtungen: Sie können das an den Rasterpunkten sogar abzählen. Wenn Ihnen das Spiel zu schnell ist, verlängern Sie die Pausenschleife in 150 (die SIMON-Anweisung PAUSE rundet leider auf volle Sekunden ab !). Übrigens ist die ganze Bahn eines Körpers, der mit stückweise gleich-

bleibenden Beschleunigungen bewegt wird, aus Parabeln und (wo die Beschleunigung null ist) Geraden zusammengesetzt, die jeweils an den Nahtstellen gemeinsame Tangenten haben. Die Parabelinnenseite ("Öffnung") ist immer die Richtung der dort gerade wirksamen Beschleunigung, in unserem Programm also eine der 8 Richtungen, die der Joystick wählen und die die ROT-Anweisung einstellen kann (diese beiden Handicaps passen also ganz gut zusammen).



```
90 REM  BESCHLEUNIGUNGSSPIEL MIT PFEILEN
91 :
92 REM JOYSTICK IN PORT 2
93 :
100 HIRES 1,0:X=20:Y=180
105 REC 0,0,319,199,1
106 FOR I=1 TO 63:FOR J=1 TO 39:PLOT I*5,J*5,1:NEXT:NEXT
110 A$="22222555666666555686868686878787878755577777779"
111 B$="22556666688887777559"
120 IF JOY=0 THEN IF X>0 AND Y>0 THEN ROT 1,1:DRAW B$,X,Y,1:GOTO 140
121 IF JOY=1 THEN W=0:        VY=VY-1
122 IF JOY=2 THEN W=1:VX=VX+1:VY=VY-1
123 IF JOY=3 THEN W=2:VX=VX+1
124 IF JOY=4 THEN W=3:VX=VX+1:VY=VY+1
125 IF JOY=5 THEN W=4:        VY=VY+1
126 IF JOY=6 THEN W=5:VX=VX-1:VY=VY+1
127 IF JOY=7 THEN W=6:VX=VX-1
128 IF JOY=8 THEN W=7:VX=VX-1:VY=VY-1
130 IF X>0 AND Y>0 THEN ROT W,1:DRAW A$,X,Y,1
140 X=X+5*VX:Y=Y+5*VY
150 FOR I=0 TO 200:NEXT:GOTO 120
```



NEUE ALPHABETE
und eigene Sonderzeichen

Bekanntlich kann man auf dem C 64 mit der SIMON-Anweisung MEM alle beliebigen Alphabete erzeugen und auf dem Bildschirm benutzen. Noch nützlicher dazu scheint mir DRAW mit ROT zu sein: Hier ist der Buchstabe weder in seiner Breite noch in seiner Höhe festgelegt, schmale und breite Buchstaben brauchen nicht in das gleiche Raster gepreßt zu werden (bei Schreibmaschinen nennt man das Proportionalschrift, beim Buchdruck ist es allgemein üblich (außer wenn ein Buch direkt mit einer Schreibmaschine gesetzt wird, wie dieses). das folgende kleine Programm reagiert bisher nur auf die Zeichen A Z I und das mit C= umgeschaltete D, das als kleines Delta erscheint. Machen Sie mit den Spielregeln für DRAW Ihr eigenes Alphabet mit den Ergänzungen, die Sie brauchen (griechische Buchstaben, Sonderzeichen für Mathematik oder andere Wissenschaften, chinesiche, arabische oder russische Buchstaben, ein Faksimile Ihrer persönlichen Unterschrift). Da die Schrift im HIRES-Modus erscheint, kann sie unmittelbar mit COPY auf den Drucker übertragen werden, wenn der Bildschirm mehr oder weniger vollgeschrieben ist.

```
700 REM  BUCHSTABEN MIT DRAW
701 :
702 REM  BITTE ALPHABETE ERGAENZEN !
703 :
790 INPUT" GROESSE   1███";G
800 HIRES 1,0:DIM A$(255),L(255)
1032 A$(32)="9":L(32)=7
1065 A$(65)="2222771155533666655777775779":L(65)=7
1073 A$(73)="7777779":L(73)=3
1090 A$(90)="5555778787878755559":L(90)=7
1172 A$(172)="0008875757577878868656559":L(172)=7
2000 GET A$:IF A$="" THEN 2000
2010 A=ASC(A$):ROT 0,G:DRAW A$(A),X,Y,1
2020 X=X+L(A)*G:IF X>320-8*G THEN X=0:Y=Y+10*G
2030 GOTO 2000
```

Die folgende Variante nutzt die vorhandenen Zeichen zum Teil über die Anweisung CHAR aus (nämlich Ziffern, große und kleine Lateinbuchstaben und einige Standardzeichen). Da man über die Tastatur noch mehr ASCII-Zeichen erreichen kann, können Sie für diese mit DRAW eigene Formen festlegen (eigene Sonderzeichen, auch z.B. den vom Lateinischen abweichenden Teil der griechischen oder russischen Buchstaben usw.). Es ist hier nur für die beiden Beispiele δ und ε (erreichbar über D und E, jeweils mit der Commodore-Taste C=) ausgeführt - vielleicht brauchen Sie ja ganz andere Zeichen ! Leider ist die variable Breite nur auf die DRAW-Zeichen anwendbar, d.h. beim gemischten Betrieb

verändern zwar alle Zeichen ihre Höhe, aber nur die mit DRAW erzeugten auch ihre Breite. Bei Größe 1 kommt das aber nicht zur Wirkung, und bei Größe 2 fällt es nur wenig auf.

```
700 REM  BUCHSTABEN MIT DRAW UND CHAR
701 :
702 REM  BITTE SONDERZEICHEN ERGAENZEN !
703 :
790 INPUT" GROESSE   1▮▮▮▮";G
800 HIRES 1,0:DIM A$(255),L(255)
1172 A$(172)="000887575757787886865655 9":L(172)=7
1177 A$(177)="2200088787558875755 9":L(172)=7:L(177)=6
2000 GET A$:IF A$="" THEN 2000
2005 A=ASC(A$)
2006 IF A>32 AND A<65 THEN CHAR X,Y,A,1,G:X=X+9:GOSUB3000:GOTO 2000
2007 IF A>64 AND A<97 THEN CHAR X,Y,A-64,1,G:X=X+9:GOSUB3000:GOTO 2000
2008 IF A>191 AND A<219 THEN CHAR X,Y,A+64,1,G:X=X+9:GOSUB3000:GOTO 2000
2010 ROT 0,G:DRAW A$(A),X,Y,1
2020 X=X+L(A)*G:GOSUB 3000
2030 GOTO 2000
3000 IF X>320-8*G THEN X=0:Y=Y+10*G
3010 RETURN
```

```
EEEeeE      EE         DdDD
1232345 67890+-£QWERTYUIOP-**ASDFGH
JKL:;ZXCVBNM,./qwertyuiopasdfghnmg
hjkl[]zxcvbnm()?
```

VIELE FARBEN TROTZ HOCHAUFLÖSUNG

Kann man nun auch mehr als 4 Farben in einer Feingrafik gleichzeitig verwenden (abgesehen von LOW COL), vielleicht sogar mit der vollen Punktauflösung von 320 x 200 ? Das geht tatsächlich, allerdings muß man hier POKE zuhilfe nehmen. Bekanntlich gibt es im Textmodus 25 Zeilen und 40 Spalten. Diese Einteilung existiert in der hochauflösenden Grafik HIRES bei SIMON's BASIC auch, und zwar für die Farben: Vorder- und Hintergrundfarbe können für jedes dieser 1000 Felder einzeln gePOKEt werden ! Wer den VC 20 mit dem Super-Expander kennt, ist dort mehr oder weniger bewußt den 200 Regionen begegnet, innerhalb derer eine mit REGION gewählte Farbe auf schon bereits gezeichnete Linien abfärbt. Dort konnte man allerdings nur die Vordergrundfarbe regional bestimmen, und die Regionen waren ziemlich grob. Beim C 64 mit SIMON's BASIC geht es zusammen mit dem hier beschriebenen Trick wesentlich feiner. Die 1000 Speicherplätze ab 48*1024 (das ist 49152 bis 50151) sind zeilenweise, von links oben angefangen, den Regionen zugeordnet, deren jede aus 8x 8 Pünktchen besteht. Da es 16 Farben gibt und diese Speicherplätze richtige ganze Bytes enthalten (was ja nicht selbstverständlich ist bei diesem Gerät), können für jeden Platz zwei Farben festgelegt werden. Und zwar sind die "oberen" für den Vordergrund (also die gesetzten Punkte) zuständig, die "unteren" für den Hintergrund. Möchte man also in die (grobe) Zeile z (von 0 bis 24) und die (grobe) Spalte s (von 0 bis 39 zulässig) die Vordergrundfarbe f1 und die Hintergrundfarbe f0 setzen, so ist zu befehlen:

```
POKE 48*1024+z*40+s,f1*16+f0
```

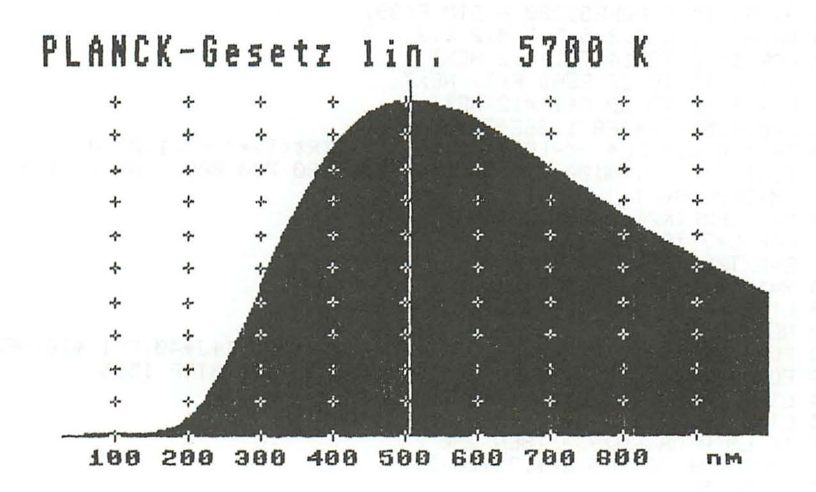
Das folgende kleine Demo-Programm schreibt Texte in den HIRES-Bildschirm bei 320 x 200 Punkten Auflösung, und zwar jeweils die Zahl f1*16+f0 und die Namen der beteiligten Farben; es läuft pausenlos immer wieder von oben weiter. Kombinationen, bei denen Sie nichts lesen können, sind offenbar auch nicht wert, gelesen zu werden.

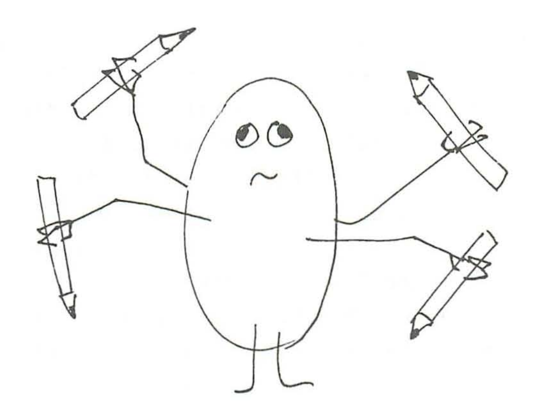
```
90 REM FARBDEMO 16*16 FARBEN MIT HIRES
91 :
100 HIRES 1,0:DIM A$(15)
110 DATA SCHWARZ,WEISS,ROT,CYAN,MAGENTA,GRUEN,BLAU,GELB
111 DATA ORANGE,BRAUN,HELLROT,DUNKELGRAU
112 DATA MITTELGRAU,HELLGRUEN,HELLBLAU,HELLGRAU
120 FOR I=0 TO 15:READ A$(I):NEXT
130 FOR I=0 TO 24
140 A=INT(16*RND(1)):B=INT(16*RND(1))
150 BLOCK 0,8*I,319,8*I+7,0
160 FOR J=0 TO 39:POKE 48*1024+40*I+J,A*16+B:NEXT
170 TEXT 5,8*I,STR$(A*16+B)+"  "+A$(A)+" AUF "+A$(B),1,1,10
180 NEXT:GOTO 130
```

Eine sinnvolle Anwendung der Kombination aus feiner Grafik und mehr als 4 Farben bietet das folgende Programm zum Strahlungsgesetz von PLANCK: Es wird eine Kurve mit der vollen Auflösung gezeichnet und nach unten hin ausgefüllt (mit LINE); das ist der mathematische Zusammenhang zwischen Intensität des Lichtes und Wellenlänge (bei der gewählten Temperatur z.B. eines Sterns oder eines Glühdrahtes, grob

```
100 REM PLANCK-GESETZ
101 :
110 INPUT"TEMPERATUR IN K     5700████████";T
120 HIRES 15,0:POKE53280,0:DIM F(39)
130 DATA 6,6,6,6,3,5,5,7,8,2,2,2,2,2
140 FOR I= 0 TO 14:F(I)=12:NEXT
141 FOR I=15 TO 27:READ F(I):NEXT
142 FOR I=28 TO 39:F(I)=12:NEXT
150 K=6.625E-34*3E8/1.38E-23*1E9
160 TEXT 0,0,"▒┐L♠/─'-|ESETZ LIN.  "+STR$(T)+" '",1,2,10
161 TEXT 20,192,"▒100 200 300 400 500 600 700 800   NM",1,1,8
170 LM=2898000/T
180 M=1/(EXP(K/T/LM)-1)/LM↑5
190 FOR L=3 TO 999 STEP 3
195 E=K/T/L:IFE>87 THEN 220
200 Y=1/(EXP(E)-1)/L↑5
210 LINE L*.32,185-Y/M*155,L*.32,185,1
220 NEXT
230 FOR I=0 TO 39:FOR J=3 TO 23:POKE 48*1024+I+J*40,F(I)*16:NEXT:NEXT
240 FOR L=100 TO 900 STEP 100:FOR Y=30 TO 185 STEP 15.5
244 LINE L*.32,Y-2,L*.32,Y+2,2
245 LINE L*.32-2,Y,L*.32+2,Y,2:NEXT:NEXT
246 IF LM<0 OR LM>999 THEN 250
247 LINE LM*.32,185,LM*.32,20,2
250 GOTO 250
```

gesagt). Nun haben diese Wellenlängen ja für uns Menschen auch etwas mit Farben zu tun: Licht einer bestimmten Wellenlänge erscheint uns in einer jeweils typischen Farbe (umgekehrt ist der Zusammenhang nicht ganz so einfach !). Es liegt daher nahe, in dem Diagramm auch diese Farben anzuzeigen, und zwar jeweils auf einer vertikalen Linie die gleiche, von links nach rechts aber (idealerweise) einen allmählichen Übergang von Violett, blau, grün, gelb, orange zu orangerot. Hier müssen wir aus zwei Gründen Kompromisse schließen: zum einen haben wir nur 16 verschiedene Farben, zum anderen können wir nur in Schritten von je 8 Rasterpunkten (also einzelnen Buchstabenbreiten des Bildschirms) die Farben wechseln. Bereiche des Spektrums, die unsere Augen nicht erfassen, sind in unserem Diagramm grau dargestellt ("unsichtbares" Licht, nämlich Ultraviolett und Ultrarot). Rasterpunkte erleichtern die Ablesung von Kurvenwerten, und das Maximum ist durch einen senkrechten Strich angezeigt. In der hier gegebenen COPY ist leider der Aspekt des Programms nicht zu sehen, für den es in diesem Abschnitt als Beispiel dient, sie mag aber trotzdem von Interesse sein.





MULTICOLOR

Halbe Auflösung, vierfarbig

Dieser Grafikmodus reduziert die waagerechte Auflösung auf die Hälfte, d.h. benutzt 160 x 200 Rasterpunkte, aber erlaubt jedem dieser immerhin noch 32000 Punkte unabhängig voneinander vier verschiedene Farben, die man mit HIRES und MULTI festlegt. Man kann also in diesem Raster 3 farbige Kurven auf einen Hintergrund zeichnen lassen, die nur in wirklichen Kreuzungspunkten aufeinander abfärben, nicht aber in der Umgebung. Mit LOW COL und seiner Umkehrung HI COL gibt SIMON uns noch 3 weitere Farben zur Auswahl (vgl. Handbuch), aber dabei gibt es ein Übergreifen der Farbe auf ein ganzes Buch-

stabenfeld (also auf ein ganzes der 1000 groben Rasterfelder). Wohl aus diesem Grunde ist LOW COL nicht auf BLOCK wirksam.

Nun braucht man die Farben in MULTI nicht unbedingt global festzulegen, sondern kann wieder auf die Speicherplätze ab 48*1024 zurückgreifen: die niedrigen 4 Bits ändern die MULTI-Farbe Nr. 2 regional (d.h. auf je einem der 1000 Felder) ab, die höheren 4 Bits (also die Vielfachen von 16) die Farbe Nr. 1. Farbe Nr. 3 ist dagegen global fest, ebenso wie 0 (der Hintergrund) und die Umrandung. Beide (normalerweise im HIRES-Befehl festgelegten) Farben kann man auch im MULTI-color-Modus mit POKE 53280,r und POKE 53281,h bestimmen. So können Sie also nebeneinander auf dem gleichen Bildschirm Regionen mit jeweils 4 verschiedenen feinaufgelösten (im 160 x 200 - Raster) Farben benutzen, von denen zwei global festliegen (Nr. 0 und 3) und zwei regional getrennt gewählt werden können (1 und 2). Dieses kleine Beispiel zeigt einige der Möglichkeiten auf:

```
100 HIRES 1,0:MULTI 2,5,6
105 POKE 53281,12:POKE 53280,10
110 BLOCK 0,0,30,199,1
111 BLOCK 40,0,70,199,2
112 BLOCK 80,0,110,199,3
120 FOR I=0 TO 239:POKE48*1024+I,7+3*16:NEXT
130 CIRCLE 50,100,45,90,1
131 CIRCLE 50,100,40,80,2
132 CIRCLE 50,100,35,70,3
200 GOTO 200
```

Der Rand ist hellrot, der Hintergrund mittelgrau. Zunächst erscheinen drei senkrechte Blocks in den MULTI-Farben 1,2,3: rot, grün, blau. Dann wird mit POKE der obere Streifen umgeändert; der Überlappungsbereich mit Farbe 1 wird dadurch cyan, der mit 2 gelb, der Rest bleibt unbeeinflußt.

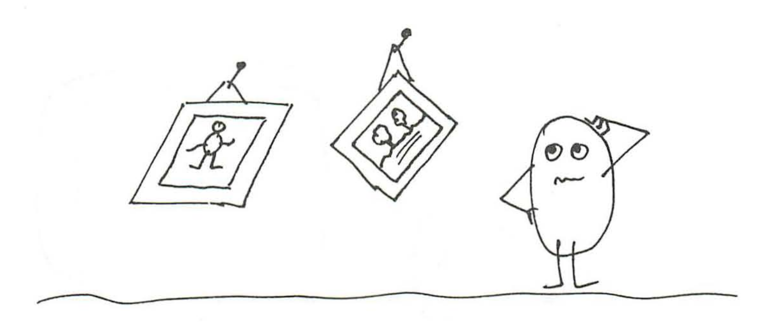
Die drei Kreise zeigen schließlich wieder die drei Farben 1 bis 3 an.

VERDREHT UND VERZERRT ?

Koordinaten-Umrechnungen

Das hört sich schlimmer an, als es ist. Hat man einen Punkt mit den Koordinaten X und Y, so kann man daraus einen neuen Punkt bestimmen mit den Koordinaten U und V und dazu die Umrechnung festlegen:

```
U=A*X+B*Y
V=C*X+D*Y
```



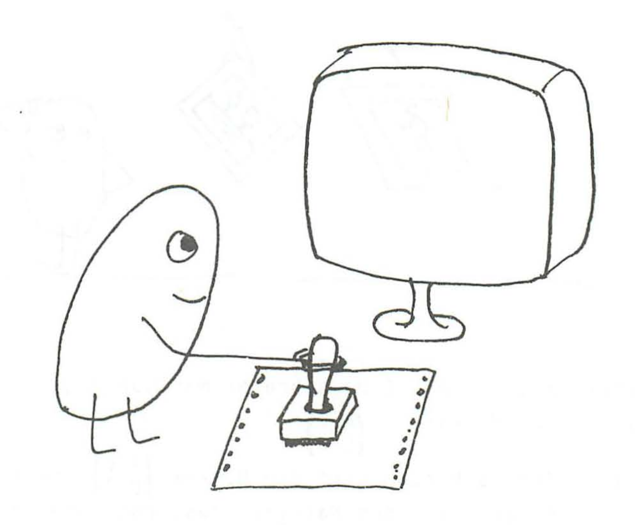
Die vier Faktoren A B C D schreibt man dabei gerne als Tabelle oder "Matrix": $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$

Wie Sie sofort sehen, ändert die Matrix $\begin{pmatrix} 1 & Ø \\ Ø & 1 \end{pmatrix}$ überhaupt nichts, sie ist unter den Matrizen das, was unter den Faktoren die Zahl 1 ist. $\begin{pmatrix} -1 & Ø \\ Ø & 1 \end{pmatrix}$ dagegen vertauscht Links und Rechts, $\begin{pmatrix} .5 & Ø \\ Ø & 1 \end{pmatrix}$ macht alles schmaler usw. Eine besonders wichtige Matrix bewirkt eine Drehung um einen Winkel W (der hier im Bogenmaß, also mit dem Wert 6.28 für die volle Umdrehung, anzugeben ist):

$$\begin{pmatrix} COS(W) & SIN(W) \\ -SIN(W) & COS(W) \end{pmatrix}$$

In dem folgenden Programm wird ein Dreieck fortlaufend um die Bildmitte gedreht und jeweils mit DRAW gezeichnet, bei Entfernung der REM-Anweisung in Zeile 16Ø auch zwischendurch gelöscht.

```
100 REM  DREHUNG EINES DREIECKS
101 :
110 HIRES 1,0
120 DATA 80,-20,-60,30,20,50
130 FOR I=1 TO 3:READ X(I),Y(I):NEXT
140 I=3:GOSUB 200:UA=U:VA=V
150 FOR I=1 TO 3:GOSUB 200:LINE UA,VA,U,V,1:UA=U:VA=V:NEXT
160 FOR J=0 TO 300:NEXT:REM     HIRES 1,0
170 W=W+.4:GOTO 140
180 :
200 REM  DREHUNG UM WINKEL W
201 :
210 U=160+X(I)*COS(W)+Y(I)*SIN(W)
220 V=100-X(I)*SIN(W)+Y(I)*COS(W)
230 RETURN
```



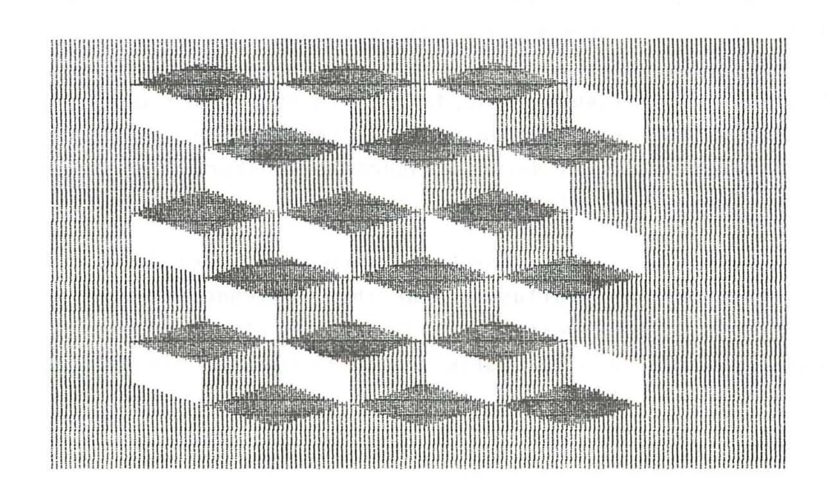
SCHWARZ AUF WEISS

COPY und HRDCPY

Wenn Sie einen Drucker VC 1525 oder VC 1515 haben, ist COPY einer der wichtigsten Befehle für die Feingrafik: er überträgt in wenigen Minuten das ganze Bild punktgetreu auf das Papier. Unterbrechen Sie diesen Vorgang bitte nicht durch die STOP-Taste, falls Sie das Programm im Speicher des C 64 noch benötigen: Ihr Rechner steigt sonst leicht aus. Bei einem Bild in MULTICOLOR tut der COPY-Befehl so,

als wären die Bits einfache Bildpunkte, d.h. er zeichnet einen "Punkt" der Farbvorwahl 3 (binär: 11) als waagerechtes Punktpaar, Das gibt bei ausgefüllten Flächen völlige Schwärzung, bei waagerechten Linien durchgezogene schmale Linien, bei senkrechten Linien doppelt breite (natürlich auch durchgezogene) Linien. Die Farbvorwahlen 1 oder 2 (binär Ø1 bzw. 1Ø) erscheinen als Einzelpunkte mit einem Zwischenraum dahinter bzw. davor, was man praktisch nicht unterscheiden kann; senkrechte Linien erscheinen dünn und durchgezogen, waagerechte fein gestrichelt, ausgefüllte Flächen werden senkrecht gestreift dargestellt. Die Farbvorwahl Ø (Hintergrund) bleibt natürlich weiß. Insgesamt haben Sie also bei der Schwarz-Weiß-Wiedergabe drei sinnvolle "Farben" zur Verfügung: Schwarz, Weiß und "Gestreift".

Ein Beispiel für dieses Verfahren sei das UMKLAPPBILD, das auf dem Bildschirm dreifarbig ist und während der Ausführung sehr schön die PAINT-Funktion zeigt. Beim Betrachten kann unser Gehirn schlagartig zwischen den beiden verschiedenen perspektivischen Deutungen wechseln. Wenn Sie die Parallelogramme lieber größer oder kleiner haben wollen, brauchen Sie nur die Größe L in Zeile 1ØØ entsprechend zu verändern.



```
90 REM UMKLAPPBILD
91 :
100 POKE 53281,2:HIRES 15,0:MULTI 6,2,7:L=10:K=L*1.6
110 FOR X=1+K TO 159-K STEP 2*K:FOR Y=2*L TO 199-2*L STEP 6*L
120 LINE X,Y-2*L,X,Y+2*L,1
121 LINE X-K,Y-L,X+K,Y+L,1
122 LINE X-K,Y+L,X+K,Y-L,1
125 NEXT:NEXT
130 FOR X=1+2*K TO 159-K STEP 2*K:FOR Y=5*L TO 199-2*L STEP 6*L
140 LINE X,Y-2*L,X,Y+2*L,1
141 LINE X-K,Y-L,X+K,Y+L,1
142 LINE X-K,Y+L,X+K,Y-L,1
145 NEXT:NEXT
150 FOR X=0 TO 159-K STEP 2*K:FOR Y=2*L TO 199-L STEP 6*L
151 PAINT X,Y,3:NEXT:NEXT
155 FOR X=K TO 159-K STEP 2*K:FOR Y=5*L TO 199-L STEP 6*L
156 PAINT X,Y,3:NEXT:NEXT
170 FOR X=K*3/2 TO 159-K STEP 2*K:FOR Y=0 TO 199-L STEP 6*L
171 PAINT X,Y,1:NEXT:NEXT
175 FOR X=K*5/2 TO 159-K STEP 2*K:FOR Y=3*L TO 199-L STEP 6*L
176 PAINT X,Y,1:NEXT:NEXT
300 GOTO 300
```

Zum Ausdrucken von Texten nimmt man normalerweise den PRINT#-Befehl oder die Hardcopy-Anweisung HRDCPY von SIMON (wenn ein auf dem Bildschirm stehender Text ausgedruckt werden soll). In beiden Fällen erscheinen die Buchstaben in der Version des Druckers (beim VC 1525 und VC 1515 in einer 6 x 7 - Matrix), die bei manchen Druckern (auch bei den genannten) von der 8 x 8 -Form des Bildschirms abweicht. Noch wichtiger ist im Zusammenhang mit Grafikbildern, die aus den Tastaturzeichen aufgebaut sind, daß PRINT# und HRDCPY zwischen den Zeilen einen Zwischenraum lassen, der zum Lesen von Texten sehr erwünscht ist, bei der "Tastaturgrafik" aber nicht. Hier hilft der Umweg über die Feingrafik mit TEXT und COPY.

Leider kann man die Breite der Buchstaben selbst bei TEXT und bei CHAR nicht beeinflussen (nur ihre Zwischenräume). Hier hilft MULTICOLOR weiter: alles wird doppelt so breit. Man kann sogar normale Feingafik und MULTICOLOR zu diesem

Zweck auf dem gleichen Bildschirm mischen. Das folgende Bild ist zwar auf dem Bildschirm teilweise kaum lesbar, ist aber in einem einzigen COPY-Vorgang gedruckt worden (darunter das zugehörige Programm):

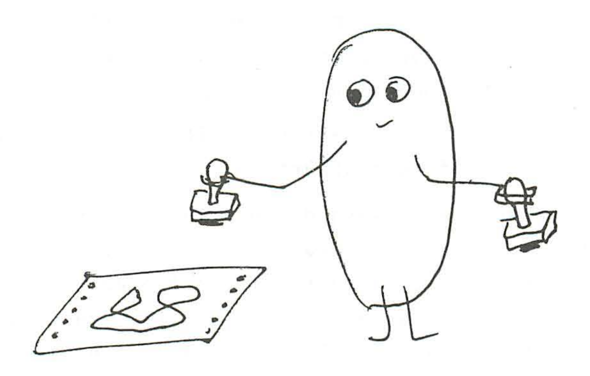
```
Mit TEXT und COPY koennen Sie die
Schrift sehr flexibel gestalten:
 feingestuftes Einruecken,
 Zeilen-Zwischenraum,
S p e r r u n g ,
doppelte Hoehe ,
und  nun  mit  MULTICOLOR:
 FARBE 3, HOEHE 1
 FARBE 2, HOEHE 2
 FARBE 1, HOEHE 3
```

```
100 HIRES 0,15:REC 0,0,319,149,1
110 TEXT 10,10,"▒╲IT |¯♦| UND ┤┌┐| KOENNEN ♦IE DIE",1,1,8
120 TEXT 10,18,"▒♦CHRIFT SEHR FLEXIBEL GESTALTEN:",1,1,8
130 TEXT 13,27,"▒FEINGESTUFTES ¯INRUECKEN,",1,1,8
140 TEXT 15,38,"▒♦EILEN-♦WISCHENRAUM,",1,1,8
150 TEXT 10,50,"▒♦PERRUNG,",1,1,12
160 TEXT 10,60,"▒DOPPELTE |OEHE ,",1,2,10
170 TEXT 10,80,"▒UND NUN MIT ╲╱L|╲┤┌┘┌─:",1,1,10
180 MULTI 0,6,6
190 TEXT 10,90,"FARBE 3, HOEHE 1",3,1,9
200 TEXT 10,100,"FARBE 2, HOEHE 2",2,2,9
210 TEXT 10,120,"FARBE 1, HOEHE 3",1,3,9
220 COPY
300 GOTO 300
```

FOTOS VOM BILDSCHIRM

Computererzeugte Grafik kann sehr schön aussehen, leider dauert das Erstellen von Feingrafik mit Formeln und Zufallsgenerator oft Stunden, und selbst das Laden eines fertigen Bildes von der Diskette ist zum bloßen Betrachten etwas umständlich. Es lohnt sich schon, hier den Fotoapparat zu benutzen. Dazu einige Tips und Vorschläge: Es ist ja klar, daß man den Bildschirm vorher reinigt, Helligkeit, Kontrast und Buntsättigung vernünftig einstellt (am besten auf eine Standard-Einstellung), und daß man störende Lichtquellen, die auf den Bildschirm scheinen, abschaltet oder abblendet. Achten Sie auch darauf, daß die Kamera genau in Höhe der Bildschirmmitte auf ihrem Stativ steht: schon kleine Neigungen des Blickes verzerren Rechtecke spürbar zu Trapezen ! Das Objektiv soll eine möglichst lange Brennweite haben, damit sich die Wölbung des Bildschirms nur wenig auswirkt; bei sehr kleinen Bildschirmen können dabei Zwischenringe nötig sein. Als Film kommen Tageslicht- oder Negativfilme in Frage, für einfarbige Bilder natürlich auch Schwarzweiß. Die Belichtungszeit sollte nicht in die Sekundenbruchteile gehen, weil sonst einzelne Streifen erkennbar verschieden oft vom Elektronenstrahl beliefert werden; es spricht nichts gegen lange Zeiten und enge Blenden, denn das Bild werden Sie ohnehin notfalls durch einen Eingriff in das Programm zum Stillstand bringen, wenn es kein reines Grafikprogramm ist. Die richtige Belichtung messen Sie am besten an einem auf mittelgrau geschalteten Bildschirm und behalten sie dann für alle Bilder bei, damit ein schwarzer Hintergrund schwarz und ein weißer weiß wird, und nicht beide grau (was die Automatik machen würde, die ja nicht weiß, ob der Hintergrund schwarz oder weiß sein soll.)

Auch wenn es Ihnen vielleicht etwas widerstrebt: es kann durchaus sinnvoll sein, die Entfernung gerade so unscharf zu stellen, daß die Rasterpunkte verschwimmen, besonders auch bei Verwendung des Zufallsgenerators zur Schaffung von Übergängen. Solche Bilder verleugnen dann etwas ihre Entstehung durch ein digitales Gerät, können aber sehr reizvoll und "echt" aussehen.



GRAFIK AUF DEM DRUCKER
auch grösser als der Bildschirm

Wenn Sie schwarzweiße Grafik auf den Drucker geben wollen, ist es beim C 64 am bequemsten, sie mit HIRES in SIMON's BASIC (oder einer anderen BASIC-Erweiterung, die feingrafik-unterstützend ist) auf den Bildschirm zeichnen zu lassen und dann (innerhalb des Programms oder hinterher im direkten Zugriff) mit COPY auszugeben. Mit dem Bildformat liegen Sie damit fest: 320 Punkte breit und 200 Punkte hoch. Mit dem Drucker können Sie jedoch 480 breit und beliebig hoch in der gleichen Feinheit zeichnen, und das auch ohne jegliche BASIC-Erweiterung; allerdings müssen Sie das Bild dann aus einzelnen Punkten aufbauen, also eigene Schleifen für Geraden oder andere Linien einbauen.

Da viele Drucker (auch die hier bevorzugt behandelten, die auch bei COPY verwendet werden können, der VC 1525 und der VC 1515) nur von oben nach unten drucken können und kein Zurück kennen, unterscheiden wir für die bildschirmunabhängige Druckergrafik drei Fälle:

1. das Bild kann von oben nach unten durch Rechnung oder Zufallseinwirkung erzeugt werden, ohne daß weiter unten befindliche Teile bekannt sind,

2. die Teile des Bildes müssen aus mathematischen Gründen in einer Reihenfolge berechnet werden, die über viele Zeilen auf- und abwärts wechseln.

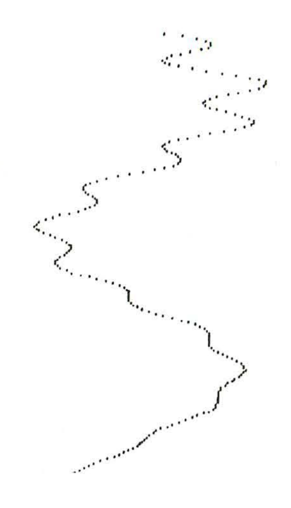
3. eine Mischform: das Wechseln hält sich in Grenzen, z.B. wenige Zeilen auf- und abwärts.

In der Gebrauchsanweisung des Druckers erfahren wir, daß im Feingrafikmodus (des Druckers !) ein Byte vom Rechner nicht als Buchstabe, sondern als Spalte aus 7 übereinanderliegenden feinen Punkten (deren 6 erst einen Buchstaben bilden) auf das Papier gibt. Dabei bedeuten die Bits Ø bis 6 von diesem Byte der Reihe nach die Besetzung der 7 Punkte von oben nach unten, das Bit Nr. 7 (mit dem Wert 128) muß dabei stets gesetzt werden. In diesem Modus druckt der Drucker Zeilen aus bis zu 48Ø solchen Säulen ohne Zwischenräume, kann also unbeschränkte Feingrafik erzeugen.

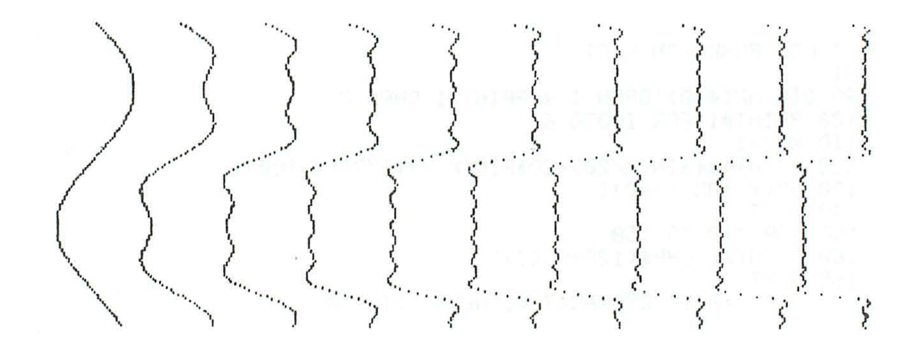
Bilder, die von oben nach unten berechnet werden können

Eine Funktion im strengen mathematischen Sinne hat zu jedem vorgegebenen "Argument" X einen eindeutig allein daraus berechenbaren Wert Y(X) (wenn es sich um eine Funktion von nur einer Variablen, also nur von einem Argument handelt).Wenn wir X in der Richtung auftragen, die zum Papiervorschub entgegengesetzt ist, kann X so lange unbeschränkt anwachsen, bis der Papierstapel erschöpft ist. Y(X) wird dann seitwärts aufgetragen. Im ersten Beispiel ist die Funktion die Überlagerung von zwei Sinuskurven mit unterschiedlicher Periode, wobei die mit der kürzeren Periode und kleineren Amplitude mit einer Exponentialfunktion abklingt (wenn Ihnen diese Bezeichnungen nichts sagen: Die Zeile 12Ø enthält diese Funktion, die Sie durch fast beliebige andere ersetzen können).

```
80 REM FUNKTION PLOT
81 :
90 DIM A%(480):OPEN 1,4:PRINT#1,CHR$(8)
100 PRINT#1:FOR I=0TO 6
110 X=X+1
120 Y=70+40*SIN(X/20)+20*SIN(X/3)*EXP(-X/100)
130 A%(Y)=A%(Y)+2↑I
140 NEXT
150 FOR J=0 TO 150
160 PRINT#1,CHR$(128+A%(J));
165 NEXT
170 FOR J=0 TO 250:A%(J)=0:NEXT:GOTO 100
```



Da der Drucker im Grafikmodus jeweils 7 übereinanderliegende Punkte als ein Byte entgegennimmt und druckt, stellt das Programm 480 solche Bytes A%() bereit. In der Schleife aus Zeile 100 wird die laufende Variable X 7-mal weitergezählt, für jedes X wird der Funktionswert Y berechnet und in Zeile 130 als einzelnes Bit in das passende Byte A%(Y) eingefügt: zuoberst in einem Siebenerblock als 1, darunter als 2, dann als 4, zuunterst schließlich als 64 (vgl. Gebrauchsanleitung des Druckers VC 1525). Sind alle 7 X-Werte dieser Serie fertig berechnet und in die Bytes eingeschrieben, wird eine ganze Zeile gedruckt (gemäß Programmzeilen 150 bis 165); der Drucker setzt dazu mehrmals an, da er im Grafikmodus nicht 80, sondern 480 Bytes für die ganze Zeile braucht und das nicht auf einmal puffern kann. In Zeile 170 werden die Bytes wieder auf 0 gesetzt, und das Spiel kann mit den nächsten 7 X-Werten weitergehen, solange das Papier reicht.



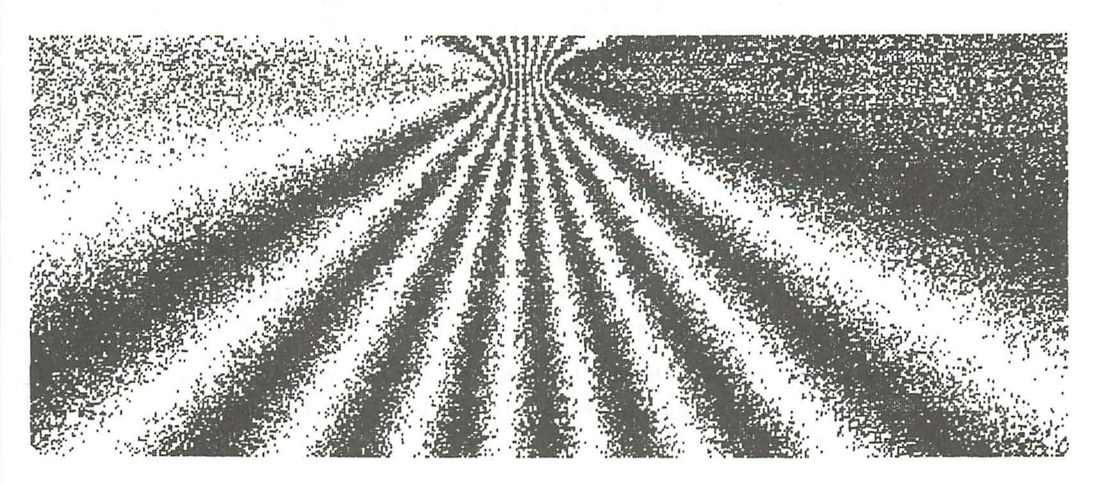
```
90 DIMA%(480):OPEN1,4:PRINT#1,CHR$(8)
100 PRINT#1:FORI=0TO6
110 X=X+.05
120 Y=30:FOR K=1 TO 19 STEP 2
121 Y=Y+17*SIN(X*K)/K+36
130 A%(Y)=A%(Y) OR 2↑I
135 NEXT
140 NEXT
150 FOR J=0 TO 479
160 PRINT#1,CHR$(128+A%(J));
165 NEXT
170 FOR J=0 TO 479:A%(J)=0:NEXT:GOTO 100
```

Im zweiten Beispiel werden gleich 10 miteinander verwandte Funktionen nebeneinander gedruckt. Anstelle der einfachen Funktionsberechnung ist eine Schleife (Zeilen 120 bis 135) getreten. Auch hier kann das Bild nach unten unbegrenzt fortgesetzt werden. Es zeigt übrigens die Fourier-Synthese einer Rechteck-Funktion aus Sinuskurven passender Amplituden und abnehmender Perioden.

Das dritte Beipiel ist ein Dichtebild, bei dem eine Funktion von X und Y abhängt und als Helligkeit dargestellt wird. Da unser Drucker nur Schwarz und Weiß unterscheiden kann, lassen wir den Computer würfeln: Je heller das Bild an der jeweiligen Stelle sein soll, um so wahrscheinlicher wird die Vorbereitung zum Drucken eines schwarzen Punktes

übersprungen (Zeile 124). Die Punkte werden in einer Doppelschleife (außen GOTO-Schleife für X, unbegrenzt; innen Y zwischen Ø und 48Ø in FOR-NEXT-Schleife) behandelt. Die in diesem Beispiel gezeigte Funktion ist das Interferenzbild von zwei kreissymmetrischen Wellen, die von zwei Punkten ausgehen und die gleiche Wellenlänge haben (wenn Ihnen der physikalische Inhalt nichts sagt: Sie können mit Funktionen wie SIN, SGN usw. fast beliebige Muster erzeugen).

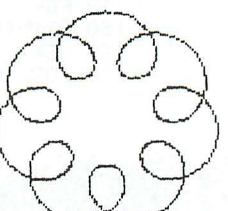
```
90 DIMA%(480):OPEN 1,4:PRINT#1,CHR$(8)
100 PRINT#1:FOR I=0 TO 6
110 X=X+1
120 FOR Y=0 TO 480
121 R1=SQR((X-20)↑2+(Y-200)↑2)
122 R2=SQR((X-20)↑2+(Y-228)↑2)
123 R=R1-R2:S=1+SIN(R)
124 IF RND(1)*2>S THEN GOTO 140
130 A%(Y)=A%(Y)+2↑I
140 NEXT:NEXT
150 FOR J=0 TO 450
160 PRINT#1,CHR$(128+A%(J));
165 NEXT
170 FOR J=0 TO 450:A%(J)=0:NEXT:GOTO 100
```



Parameter-Darstellungen

Hinter diesem Mathematiker-Chinesisch verbirgt sich ein Prinzip, eine Kurve zu zeichnen, die in X- und in Y-Richtung gleichermaßen wandert, und zwar durchaus vor- und rückwärts. Zur Berechnung von X und Y muß man dabei auf eine andere Größe zurückgreifen, die nachher im Bild gar nicht mehr zu sehen ist, die aber entlang der Kurve mehr oder weniger gleichmäßig fortschreitet: anschaulich auf die Zeit, die während des Zeichnens abläuft; allgemein wird diese Größe in diesem Zusammenhang "Parameter" (="Nebengröße") genannt. Da unser Drucker leider nicht das Papier abwechselnd vor- und zurückschieben kann, helfen wir uns mit einem Vorrat an Bytes, der für das ganze Bild ausreicht. Unsere Kurve kann dann kreuz und quer laufen; ihre Punkte werden mit OR in die passenden Bits gesteckt (Zeilen 140 bis 150). Ist die Kurve fertig im Speicher, wird das ganze Bild auf

```
80 REM PARAMETER-PLOT
81 :
100 DIM A%(100,14)
110 FOR T=0 TO π*80 STEP .3
120 X=50+35*SIN(T/40)+14*SIN(T/5)
130 Y=50+35*COS(T/40)+14*COS(T/5)
140 Y1=INT(Y/7):Y2=INT(Y-7*Y1)
150 A%(X,Y1)=(A%(X,Y1) OR 2↑Y2)
160 NEXT
200 OPEN1,4:PRINT#1,CHR$(8)
210 FORY1=0TO14:FORX=0TO100
220 PRINT#1,CHR$(128+A%(X,Y1));
230 NEXT:PRINT#1:NEXT
```



einmal gedruckt. In unserem Beispiel ist es eine Epizykloide, eine Kurve also, die beim Wandern eines Kreises auf einem größeren Kreis entstehen kann.

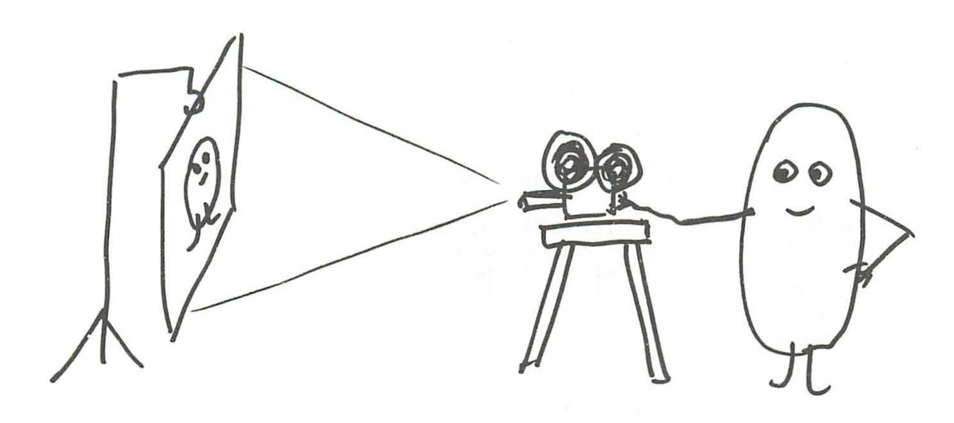
Dieses Verfahren ist hinsichtlich der Kurven völlig frei, ist aber im Format durch den Speicherplatz des Computers beschränkt.

Eine Mischform

In speziellen Fällen kann es vorkommen, daß man ein unbeschränkt langes Bild drucken möchte, bei dem trotzdem nicht alle Punkte in der genauen Reihenfolge des Papiervorschubes berechnet werden können. Das Beispiel zeigt die Reflexion einer transversalen Welle zwischen einem losen und einem festen Ende (falls Ihnen das nichts sagt, betrachten Sie es als schönes Tapetenmuster). Die einzelnen Kurven schwingen nach oben und unten durchaus weiter als um eine Druckzeile. Auf der anderen Seite kann der C 64 unmöglich alle Bytes speichern für das ganze Bild.Der Trick besteht nun darin, jeweils drei aufeinanderfolgende Druckzeilen im Computer zu speichern und während der Rechnung in sie Bits einzufügen (also Bildpunkte). Dann wird die oberste von ihnen gedruckt (Zeile 300), und die anderen beiden Zeilen werden im Variablenspeicher um 1 verschoben (Zeile 310) und geleert (320). Durch die mit K indizierte Schleife in 300 wird das Bild um den Faktor 5 seitwärts gestreckt. Die

```
90 REM TRANSV. REFLEXION
91 DIMG%(2,80):OPEN 1,4:PRINT#1,CHR$(8)
100 DIM V(80),Y(80):DT=.3:D=1
110 FOR I=0 TO 80:Y(I)=10*EXP(-(I-20)↑2/30):NEXT
200 FOR I=0 TO 80
201 Y=Y(I)+10.5:Y1=INT(Y/7):Y2=INT(Y-7*Y1)
202 G%(Y1,I)=G%(Y1,I)OR2↑Y2
203 NEXT
210 FOR I=0 TO 80:Y(I)=Y(I)+V(I)*DT:NEXT
220 FOR I=1 TO 80
230 DL=0:IF I>0 THEN DL=Y(I)-Y(I-1)
240 DR=0:IF I<80 THEN DR=Y(I+1)-Y(I)
250 V(I)=V(I)+D*DT*(DR-DL):NEXT
260 P=P+1:IF P=7 THEN P=0:GOTO 300
280 GOTO 210
300 FOR J=0 TO 79:FOR K=0 TO 4:PRINT#1,CHR$(128+G%(0,J));:NEXT
310 G%(0,J)=G%(1,J):G%(1,J)=G%(2,J)
320 G%(2,J)=0:NEXT:PRINT#1:GOTO 200
```

Zeilen 100 bis 280 beschreiben im wesentlichen den physikalischen Inhalt des Programms (der hier nicht weiter erläutert werden soll); 201 bis 202 speichern die zu zeichnenden Punkte ab.



MINI-KINTOPP

Bewegte Bilder

Der Speicherplatz 53272 schaltet nicht nur zwischen den beiden Zeichensätzen (21 für Großbuchstaben und volle Tastaturgrafik, 23 für Klein- und Großbuchstaben mit eingeschränkter Tastaturgrafik) um, sondern legt auch mit seinen oberen 4 Bits fest, wo innerhalb der (normalerweise ersten) 16 kB der Bildschirmspeicher abgefragt werden soll. Bekanntlich ist das normalerweise ab 1024: das ist 1 x 1024; darum steckt in der 21 oder in der 23 die 16 einmal. Erhöht man den Inhalt um 16, so verlagert sich der Bildschirmspeicher um 1024. In dem folgenden Programm werden erst 8 kB leergeräumt (da in BASIC, dauert das etwas !), dann werden mit der Tastaturgrafik drei verschiedene bewegliche Objekte gemalt: ein von links nach rechts schwingender kleiner Ball, eine von links nach rechts wandernde Welle, und unten eine Saite, die mit mehreren Knotenpunkten schwingt (sogenannte stehende Welle). Die Bilder werden für 8 aufeinanderfolgende Zeitpunkte (nach denen sich alles wiederholen soll) in 8 Bildschirmspeicher gesetzt (auch das dauert einige Minuten). Was dann aber wirklich schnell geht, ist das Wechseln zwischen den 8 Bildern: Sie sehen 15 Bilder aus je 1000 Zeichen pro Sekunde.

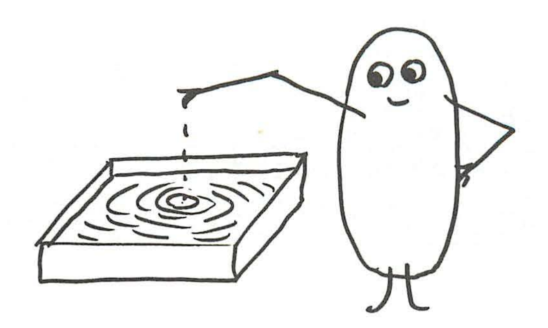
```
70 REM  KINO MIT 8 BILDSCHIRMINHALTEN
71 :
72 REM  CA. 10 MIN VORLAUF
73 :
74 REM  LEERTASTE KURZ ODER LANGE DRUECKEN
75 :
76 REM  NEUSTART MIT "GOTO 140"
77 :
80 DATA 32,99,119,120,226,249,239,228
81 DATA 99,69,68,67,64,70,82,100
82 FOR I=0 TO 7:READ GR(I):NEXT
83 FOR I=0 TO 7:READ GZ(I):NEXT
90 FOR I=0 TO 8191:POKE 8192+I,32:PRINTI:NEXT
100 FOR T=0 TO 7
110 FOR X=0 TO 39:PRINTT,X
120 Y=7-3*COS(X*π/8-T*π/4)
121 Z=15-3*COS(X*π/8)*COS(T*π/4)
125 YG=INT(Y):YF=(Y-YG)*8
126 ZG=INT(Z):ZF=(Z-ZG)*8
130 FOR H=0 TO YG:POKE8192+1024*T+H*40+X,160:NEXT
135 POKE 8192+1024*T+(YG+1)*40+X,GR(YF)
136 POKE 8192+1024*T+(ZG+1)*40+X,GZ(ZF):NEXT
137 XX=20+18*SIN(T*π/4)
138 POKE 8192+1024*T+XX,209
139 NEXT
140 T=0:FOR I=0 TO 1024:POKE 55296+I,6:NEXT
200 T=T+1:IFT=8 THEN T=0
220 POKE 53272,133+T*16
230 GETA$:IFA$=""THEN230
240 GOTO 200
```

DIE FLOPPY ALS FILM

Das zweite Beispiel ist ebenfalls aus der Physik (der Grund ist außer meinem Beruf einfach der, daß man physikalische Dinge auch dann, wenn sie kompliziert aussehen, mit einer einfachen Formel erfassen kann, also mit einem kurzen Programm etwas Eindrucksvolles zeigen kann !). Hier geht es um Wellen in einem zweidimensionalen Gebiet, wie man sie im Analogieexperiment in der Schule in einer Wellenwanne vorführt. Nehmen wir an, daß ein Punkt durch einen äußeren Einfluß in Schwingungen mit gleichbleibender Frequenz versetzt wird. Wenn er von einem Medium umgeben ist, das Wellen transportieren kann, werden rundherum alle Punkte auch von diesen Schwingungen erfaßt, aber mit einer Zeitversetzung, die einfach mit dem Abstand vom Erregungspunkt zunimmt. Der

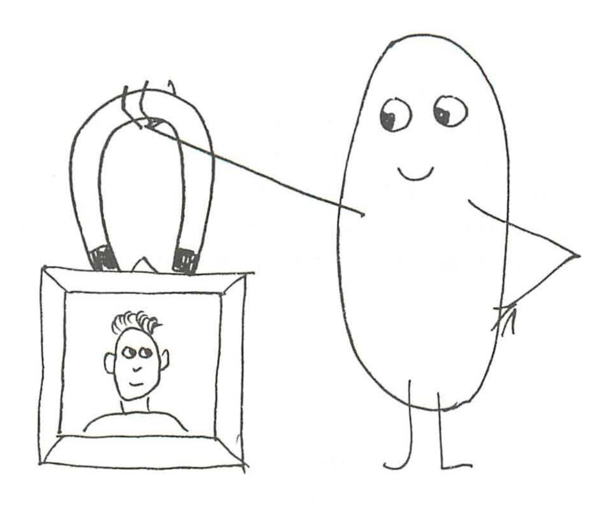
ganze raumzeitliche Vorgang ist dann eine Welle, und zwar in diesem einfachen Fall eine konzentrische Welle. Unser Programm zeigt sie, wenn Sie X1 und X2 in Zeile 1ØØØ auf den gleichen Wert setzen.
Interessant wird es nun aber, wenn an zwei Stellen zugleich solche Wellen erregt werden. Überall addieren sich beide einfach (vornehm in der Sprache der Physiker: Superposition). Am besten sehen Sie sich das an. Leider hat das Ganze einen Haken: die Berechnung der Bilder (8 Zeitpunkte, jeweils 4ØØØ Punkte in Viertelgrafik) dauert rund 100 Minuten. Darum ist das Programm so geschrieben, daß es bei angeschlossener Diskettenstation die Rechenergebnisse auf der Diskette als sequentielle Datei ablegt (bei mehreren Aufnahmen auf der gleichen Diskette bitte den Namen in Zeilen 1Ø7Ø vor dem ersten Komma ändern !). Das Wiedergabeprogramm kann dann in nur 5 Minuten den Film wieder zum Laufen bringen, auch wenn der Inhalt des Rechners inzwischen gelöscht wurde.

Da in diesem Film nur 4ØØØ Punkte räumlich aufgelöst werden, ist es von verblüffender Wirkung, wenn man den Bildschirm mit entsprechend kleinerer Sehschärfe ansieht: Kurzsichtige nehmen dazu ihre Brille ab, auch eine weiße Plastictüte unmittelbar vor dem Bildschirm setzt die räumliche Auflösung auf die Zahl der wirklich berechneten Punkte herab und täuscht damit ein Bild mit gleichmäßigen Übergängen vor.



```
90 REM WELLENWANNE AUFNAHME C 64 BASIC 100 MIN VORLAUF
95 POKE53281,1
100 DIM G%(16),GI%(256)
101 FOR I=0 TO 15:READ G%(I):GI%(G%(I))=I:NEXT
110 DATA 32,126,124,226,123,97,255,236,108,127,225,251,98,252,254,160
900 FOR I=0 TO 999:FOR J=0 TO 7:POKE 8192+J*1024+I,32:NEXT
910 POKE 55296+I,6:NEXT
950 POKE 53272,133
1000 X1=30:X2=50:Y1=25:Y2=25
1010 FOR X=0 TO 79:FOR Y=0 TO 49
1020 R1=SQR((X-X1)↑2+(Y-Y1)↑2*1.3)
1021 R2=SQR((X-X2)↑2+(Y-Y2)↑2*1.3)
1035 FOR T=0 TO 7:P0=2↑13+T*1024+960
1040 A1=COS(R1-T*π/4)
1041 A2=COS(R2-T*π/4)
1042 A=A1+A2
1050 IF A>0 THEN GOSUB 9000
1060 NEXT:NEXT:NEXT
1070 OPEN 2,8,2,"WAN3,S,W"
1071 FOR I=0 TO 8191:PRINT#2,PEEK(8192+I):NEXT
1072 CLOSE 2
1100 FOR T=0 TO 7:POKE 53272,133+16*T:GOSUB 2000:NEXT:GOTO 1100
2000 IF PEEK(203)<>64 THEN FOR TT=0 TO 2↑(7 AND PEEK(203)):NEXT
2001 RETURN
9000 XI=INT(X/2):YI=INT(Y/2)
9010 GR=(1-(2*XI<>INT(X)))*(1-3*(2*YI=INT(Y)))
9020 POKE P0+XI-40*YI,G%(GR OR GI%(PEEK(P0+XI-40*YI)))
9030 RETURN
```

```
1000 REM WANNE WIEDERGABE
1020 POKE 53281,1
1060 FOR I=0 TO 1023:POKE 55296+I,0:NEXT
1070 OPEN 2,8,2,"WAN3,S,R"
1071 FOR I=0 TO 8191:INPUT#2,A:POKE 8192+I,A:NEXT
1072 CLOSE 2
1100 FOR T=0 TO 7:POKE 53272,133+16*T:GOSUB 2000:NEXT:GOTO 1100
2000 IF PEEK(203)<>64 THEN FOR TT=0 TO(PEEK(203) AND 7)*8:NEXT
2001 RETURN
```



DIE FLOPPY ALS FOTOALBUM

64000 Punkte auf Diskette

Den Trick des vorigen Abschnitts kann man natürlich auch auf einzelne Bilder anwenden, obwohl es sich dann kaum lohnt; interessanter wird es schon bei der HIRES-Grafik, mit oder ohne MULTIcolor. Das folgende Programm dazu verdanke ich Herrn W. Mexner, Duisburg, der dabei ein kleines Maschinensprachenprogramm aus dem Data-Becker-Buch "64 Tips & Tricks" benutzt hat.Dieses ist nämlich nötig, um die Speicherplätze mit dem HIRES-Bild auszulesen: mit PEEK geht das wegen der Mehrfachbelegung gewisser Speicherbereiche beim C 64 leider nicht. Das Beispiel enthält in den Zeilen 200 bis 240 ein Grafikprogramm, das Sie durch ein anderes ersetzen können. Am besten geben Sie der Aufzeichnung auf der Diskette dann auch einen entsprechenden Namen (Zeile 260 bzw. 120 bei der Wiedergabe). Die "Inversionen" werden selbst schneller aus-

geführt als die Wiedergabe des abgespeicherten Bildes von der Diskette dauert; es eignet sich also gut zur Erprobung des Verfahrens, ist aber sonst nicht das lohnende Objekt: Sinnvoll wird es bei Programmen, deren Bilderzeugung eine ganze Nacht dauert: die Aufzeichnung können Sie dann in wenigen Minuten auf den Bildschirm zaubern !

```
100 REM HIRES ABSPEICHERN
110 :
120 POKE 785,60:POKE 786,3
130 FOR I=828 TO 875:READ A:POKE I,A:NEXT
140 DATA 165,20,128,165,21,128,32,247,183,165,1,72
150 DATA 165,21,201,208,144,7,201,224,176,3,169
160 DATA 49,44,169,52,120,133,1,160,0,177,20,168
170 DATA 104,133,1,88,104,133,21,104,133,20,76,162,179
180 N=7
190 :
200 HIRES 1,0
210 FOR I=0 TO 199
220 LINE 319,I,0,199-I,2:NEXT
230 FOR I=0 TO 319
240 LINE I,0,319-I,199,2:NEXT
250 :
260 OPEN 2,8,6,"INVERSION,SEQ,W"
270 FOR I=57344 TO 65344
280 PRINT#2,USR(I):NEXT
290 CLOSE 2
```

```
100 REM WIEDERGABE HIRES
110 HIRES1,0
120 OPEN2,8,6,"INVERSION,SEQ,R"
130 FORI=57344TO65344
140 INPUT#2,A:POKEI,A:NEXT
150 CLOSE2
160 GOTO60
```

GEISTERHAENDE

Sprite-Grafik

Neben dem Synthesizer ist die Sprite-Grafik die herausragende Spezialität des C 64. Sie ist offenbar für Telespiele entwickelt, aber auch sonst recht gut brauchbar. Ein Sprite (was wörtlich "Kobold" oder "Geist" heißt) ist hier ein kleines Bild, das sehr komfortabel auf dem Bildschirm bewegt werden kann, ohne daß die sonstigen Inhalte dadurch dauerhaft verändert würden: das Sprite wird also keineswegs in das vorhandene Bild geschrieben, sondern hat seinen eigenen Speicherbereich. Ist es eingeschaltet, so erhält das Fernsehbild eine Mischung aus dem normalen (Text- oder Grafik-)Bild und den Sprites (bis zu acht Stück gleichzeitig). Sowohl beim Bild als auch beim Sprite gibt es eine Hintergrundfarbe (es sei denn, es wäre alles vollgeschrieben). Bei hochauflösender Grafik gibt es eine Vordergrundfarbe, bei Multicolor deren drei (auch das gilt für die Sprites genau so wie für die Text- oder Grafikbelegung des Bildschirms). Die Hintergrundfarbe ist für alle Sprites und den Bildschirm stets gleich (sie steht im Speicherplatz 53281), die Vordergrundfarben sind dagegen unabhängig voneinander. An den Stellen, an denen Sprite und normaler Bildinhalt oder eins von beiden die Hintergrundfarbe hat, ist alles einfach. Was ist aber zu sehen, wo Bild und Sprite eine Vordergrundfarbe zeigen "wollen" ? Nehmen wir an, mit normaler Feingrafik ist ein Haus gemalt, das Sprite zeigt dagegen ein Auto. Nun bewegt sich das Auto über das Haus hinweg, d.h. besetzt vorübergehend den gleichen Platz auf dem Schirm. Soll nun das Haus verdeckt werden, oder soll das Auto scheinbar hinter dem Haus vorbeifahren ? Beides läßt sich machen: wir können dem Sprite (und zwar jedem einzeln) Priorität vor dem normalen Bildschirminhalt einräumen oder verweigern.

```
90 REM  EIN BLAUES AUTO FAEHRT UM DAS ROTE HAUS
91 :
92 REM  BEACHTEN SIE BESONDERS DIE FENSTER !
93 :
94 REM  ZEITLUPE MIT SHIFT (-LOCK)
95 :
100 X=100:HIRES 10,15
110 BLOCK 100,120,200,150,1
120 FOR I=0 TO 2:BLOCK 110+30*I,125,130+30*I,140,0
121 BLOCK 120+30*I,120,121+30*I,140,1:NEXT
122 BLOCK 100,131,200,132,1
130 LINE 90,120,150,90,1
131 LINE 210,120,150,90,1
132 LINE 90,120,210,120,1
133 PAINT 150,100,1
150 DESIGN 0,32*64+48*1024
151 @..........BBBBBBBBB.....
152 @.........B.....B...B....
153 @........B...BB.B....B...
154 @.......B...BBB.B.....B..
155 @......B..B..BB.B......B.
156 @...BBBBBBBBBBBBBBBBBBBB.
157 @.BBBBBBBBBBBBBBBBBBBBBBB
158 @BBBBBBBBBBBBBBBBBBBBBBBB
159 @BBBBBBBBBBBBBBBBBBBBBBBB
160 @.BBBBBBBBBBBBBBBBBBBBBB.
161 @...BBBBB.......BBBBB....
162 @....BBB.........BBB.....
163 @........................
164 @........................
165 @........................
166 @........................
167 @........................
168 @........................
169 @........................
170 @........................
171 @........................
175 DESIGN 0,33*64+48*1024
176 @.....BBBBBBBBB..........
177 @....B...B.....B.........
178 @...B....B.BB...B........
179 @..B.....B.BBB...B.......
180 @.B......B.BB..B..B......
181 @.BBBBBBBBBBBBBBBBBBBB...
182 @BBBBBBBBBBBBBBBBBBBBBBB.
183 @BBBBBBBBBBBBBBBBBBBBBBBB
184 @BBBBBBBBBBBBBBBBBBBBBBBB
185 @.BBBBBBBBBBBBBBBBBBBBBB.
186 @...BBBBB........BBBBB...
187 @....BBB..........BBB....
```

```
188 @........................
189 @........................
190 @........................
191 @........................
192 @........................
193 @........................
194 @........................
195 @........................
196 @........................
200 MOB SET 0,33,6,1,0
201 RLOCMOB 0,X,177,3,1
202 IF PEEK(653)>0 THEN FOR I=0 TO 100:NEXT
203 IF X<330 THEN X=X+2:GOTO 201
205 MOB SET 0,32,6,0,0
206 RLOCMOB 0,X,177,3,1
207 IF PEEK(653)>0 THEN FOR I=0 TO 100:NEXT
208 IF X>0 THEN X=X-2:GOTO 206
220 GOTO 200
```

Das gilt merkwürdigerweise jedoch nicht im Multicolor-Modus des Hauptbildes: dort haben die Sprites immer "Vorfahrt". Überlappen sich zwei Sprites, so hat stets das mit der niedrigeren Nummer die Priorität vor dem anderen. Sollen sich zwei Sprites abwechselnd so begegenen, daß mal das eine, mal das andere vorne ist, muß man die Zuordnung zwischen ihren Nummern und ihren Gestalten (d.h. Speicherbereichen) jeweils umkehren.

Auf dem Bildschirm können gleichzeitig bis zu 8 Sprites (Nummern Ø bis 7) erscheinen, die Zahl der verfügbaren Gestalten ist aber nur durch den Speicherplatz beschränkt. Formänderungen einer Figur kann man erreichen, indem man der gleichen Sprite-Nummer bei unveränderten Koordinaten (also Ort auf dem Bildschirm) nacheinander verschiedene Speicherbereiche (in denen die Gestalt festgelegt ist) zuweist.

Jedes Sprite enthält 504 bit Information, aufgeteilt auf 21 Rasterzeilen. Bei nur einer Sprite-Vordergrundfarbe bedeutet das 24 Punkte nebeneinander, bei Multicolor 12 Punkte, die einzeln zwischen Hintergrund und drei Vorder-

grundfarben zu wählen sind. Ein Sprite entspricht also in der Größe fast 3 x 3 Buchstaben (in der Höhe nicht ganz). Man kann es waagerecht und/oder senkrecht auf das Doppelte spreizen, wobei einfach jeder Punkt doppelt oder gar vierfach erscheint, die Auflösung also nicht größer wird. Ob ein Sprite mit einem anderen oder mit einem Bildelement der normalen Grafik zusammenstößt (etwa bei einer Bewegung; und zwar geht es dabei nur um die "undurchsichtigen Teile", also die Vordergrundpunkte), kann abgefragt werden.

Die erwähnten 504 bit sind 63 Bytes. Diese füllen (fast) einen Speicher-"Block" von 64 Bytes. Dieser kann überall im verfügbaren Speicherbereich liegen, aber aus Gründen des Zugriffs stets beginnend mit einem Vielfachen von 64. Jeder derartige Speicherblock kann als Bildinhalt eines Sprites verwendet werden, und es können durchaus wesentlich mehr als acht verschiedene sein: Sie können sich also gewissermaßen einen ganzen Fundus von Bildern bereitlegen, von denen bis zu acht gleichzeitig als Sprites durch das Bild geistern

Wie wird die Sprite-Figur aufgebaut ?

Für das einfarbige Sprite haben wir 24 Spalten und 21 Zeilen, für das Multicolor-Sprite 12 entsprechend doppelt so breite Spalten, aber jeweils zwei Bits für jedes Pünktchen zur Festlegung der Farbe (2 bit erlauben die Auswahl aus 4). Auf jeden Fall gibt es pro Zeile 24 Bits, also 3 Bytes. Die Anordnung im C 64 ist nun so gewählt, daß diese drei direkt hintereinander liegen, dann die drei der nächsten Zeile, jeweils von links nach rechts, insgesamt von oben nach unten. Soll ein Bild in Hochauflösung (nicht Multicolor) erschei-

nen, bedeutet eine Ø als Bit eine durchsichtige Stelle, eine 1 die Vordergrundfarbe des Sprites. Bei Multicolor sind die Bits immer paarweise zusammenzufassen: ØØ ist durchsichtig, Ø1, 1Ø und 11 sind die drei Farben (deren Bedeutung an anderer Stelle vereinbart wird). Man muß nun in irgendeiner Form diese Bits zu Bytes zusammenrechnen, also in jeder Gruppe von 8 Bits für eine 1 ganz links 128 setzen, für die nächste Stelle 64 usw. Das Ergebnis muß dann an die passende Stelle gePOKEt werden. Bei Multicolor können wir auch, statt über die einzelnen Bits zu rechnen, vier Stellen mit jeweils den Zahlen von Ø bis 3 unterscheiden; sie sind je nach der Position (von links nach rechts) zu multiplizieren mit:

64	16	4	1	, also:
128	32	8	2	bzw.
192	48	12	3.	

Bei einfachen Sprite-Formen kann es durchaus sinnvoll sein, solche Rechnungen selbst zu machen und die Ergebnisse dem Programm per DATA-Zeilen, READ und POKE einzuverleiben. Weniger platzsparend, aber anschaulicher und bei komplizierteren Programmen bequemer ist die DESIGN-Anweisung in SIMON's BASIC. Soll das Sprite aber nicht vom Programmierer, sondern vom Benutzer des Programms entworfen werden, geht man über INPUT. Schließlich kann es noch reizvoll sein, ein Sprite aufgrund einer mathematischen Festlegung vom Programm berechnen zu lassen. Dazu folgen einige Beispiele.

Sprites per INPUT

Dieses Hilfsprogramm dient (etwa als Teil- oder Unterprogramm) dazu, ein einfarbiges Sprite mit INPUT einzugeben. Dabei ist angenommen, daß Sie die Figur bereits fertig auf kariertes Papier gezeichnet haben, also nicht mehr auf dem Bildschirm korrigieren müssen. Leere, also durchsichtige Stellen können Sie mit einem Punkt oder der Leertaste eingeben, ganze leere Zeilen einfach durch

leere Eingabe (also Return-Taste sofort drücken). In Zeile 136 sind die Tasten auf Dauerfunktion geschaltet (mit POKE 650,0 wird das rückgängig gemacht). Wenn Sie mehr als 24 Zeichen für eine Zeile eingeben, wird der Rest ignoriert. Alle Zeichen außer Punkt und Leertaste setzen einen Punkt in das Sprite. Nach der Eingabe aller 21 Zeilen dauert es etwas, bis das BASIC-Programm fertig ist. Es zeichnet dann das Sprite und vergrößert es beim Drücken der SHIFT-Taste (Zeile 350 als Fangschleife).

Wenn Sie das Programm weiter ausbauen wollen, brauchen Sie weitere Einzelheiten: AA in Zeile 130 ist die Start-Adresse des Speicherbereichs für die Sprite-Form; sie ist stets das 64-fache der Block-Nummer, wie sie in Zeile 310 auftritt (hier: 128). Der Kern des Programms ist der Teil von 180 bis 250, besonders 215 bis 240: Hier werden die X-Werte auf die drei nebeneinanderliegenden Bytes verteilt.

```
80 REM SPRITE PER INPUT
81 :
83 REM VERGROESSERUNG MIT SHIFT
84 :
90 DIM A$(21)
91 POKE 53248+32,6:POKE 53248+33,6:PRINT"□■"
130 AA=2↑13
135 FOR I=AA TO AA+63:POKE I,0:NEXT
136 POKE 650,128
138 PRINT"□    123456789012345678901234"
139 PRINT"       ◤         ▼         ◥"
140 FOR I=1 TO 21:I$=RIGHT$(STR$(I),2)
150 PRINT I$;:INPUT A$(I):NEXT
160 :
180 FOR Y=1 TO 21
181 IF LEN(A$(Y))<24 THEN A$(Y)=A$(Y)+"........................"
182 FOR X=0 TO 23
190 Z$=MID$(A$(Y),X+1,1)
200 IF Z$="." OR Z$=" " THEN 250
215 X1=INT(X/8):X2=X-8*X1
230 A1=AA+3*Y+X1
240 A2=PEEK(A1):A3=2↑(7-X2):A4=A2 OR A3:POKE A1,A4
250 NEXT:NEXT
260 :
300 V=53248:PRINT"□"
310 POKE 2040,128
320 POKE V+21,1
330 POKE V+39,1
340 POKE V+0,100:POKE V+1,100
350 POKE V+23,PEEK(653):POKE V+29,PEEK(653):GOTO 350
```

Dasselbe mit MULTIcolor

Das Multicolor-Sprite hat außer durchsichtigen Punkten bis zu drei verschiedene Farben, dafür hat es nur halb so viele doppelt so breite "Punkte". Unser Programm erwartet bei der Eingabe einfach die Zahlen 1 bis 3 für die Farben und deutet alle anderen Zeichen als durchsichtige Punkte. Die Prozedur des Einsortierens der Bits in die Bytes ist hier etwas anders, da die Zahlen von Ø bis 3 zweistellige Binärzahlen sind und als Ganzes eingebaut werden müssen (in die 8 Bits eines jeden Bytes passen darum gerade vier Multicolor-Punkte). Die Farben werden so zugeteilt:

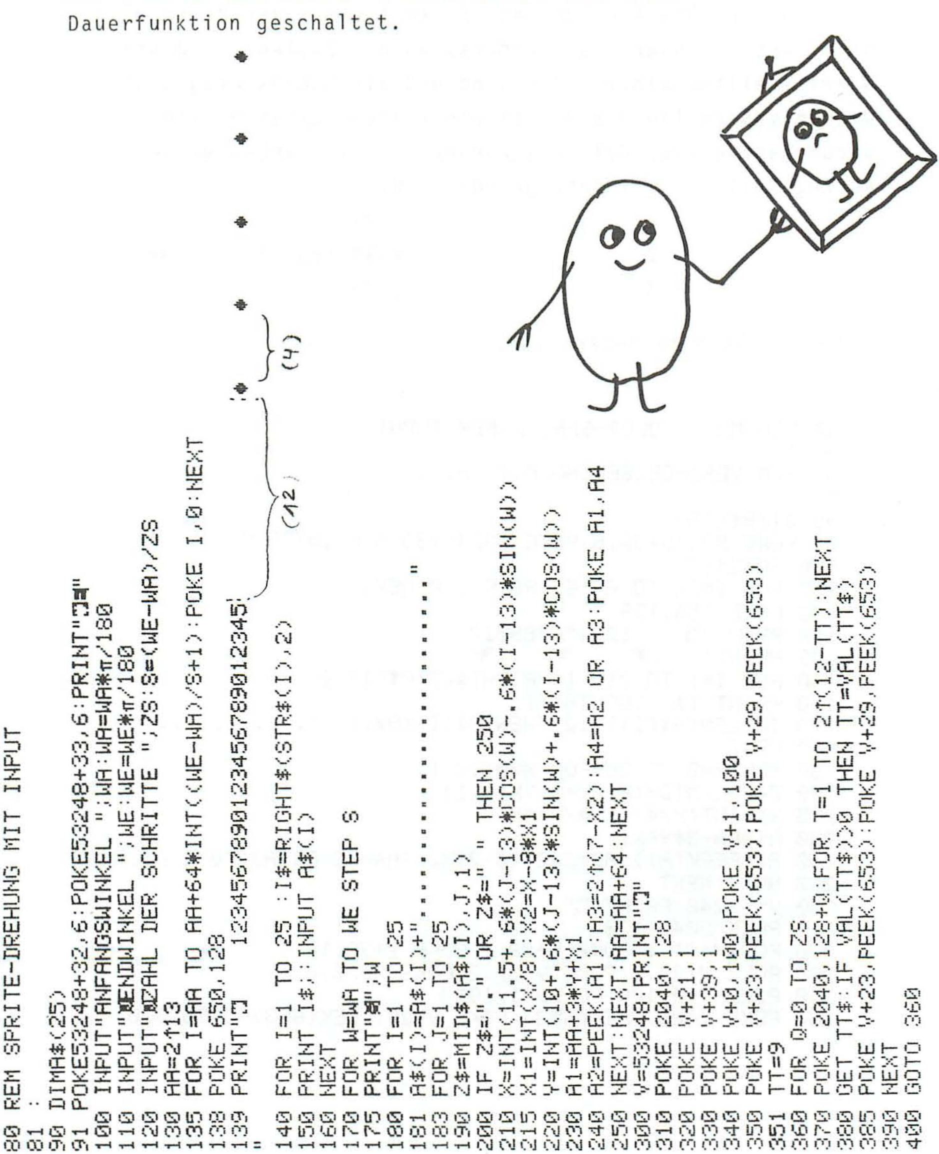
Ø (Hintergrund)	V+33	
1	V+37	
2	V+39	(für Sprite Nr. Ø)
3	V+38	

wobei V=53248 zu setzen ist.

```
80 REM MULTICOLOR-SPRITE PER INPUT
81 :
82 REM VERGROESSERUNG MIT SHIFT
83 :
90 DIMA$(25)
91 POKE 53248+32,6:POKE 53248+33,6:PRINT"▯▮"
130 AA=2↑13
135 FOR I=AA TO AA+63:POKE I,0:NEXT
136 POKE 650,128
138 PRINT"▯    123456789012"
139 PRINT"      ◤     ▼     ◥"
140 FOR I=1 TO 21 :I$=RIGHT$(STR$(I),2)
150 PRINT I$;:INPUTA$(I)
155 IF LEN(A$(I))<12 THEN A$(I)=A$(I)+"............"
160 NEXT
180 FOR Y=0 TO 20:FOR X=0 TO 11
190 Z=VAL(MID$(A$(Y+1),X+1,1))
215 X1=INT(X/4):X2=X-4*X1
230 A1=AA+3*Y+X1
240 A2=PEEK(A1):A3=Z*2↑(6-2*X2):A4=A2 OR A3:POKE A1,A4
250 NEXT:NEXT
300 V=53248:PRINT"▯"
310 POKE 2040,128
320 POKEV+21,1:POKE V+28,1:POKE V+33,12
330 POKE V+39,7:POKE V+37,2:POKE V+38,6
340 POKE V+0,100:POKEV+1,100
350 POKE V+23,PEEK(653):POKE V+29,PEEK(653):GOTO 350
```

ROTIERENDE SPRITES

Manchmal möchte man ein Sprite auch noch drehen, und zwar nicht unbedingt nur um Vielfache von 45° oder gar nur solche von 90° (was ja noch relativ einfach geht). Das folgende Programm nimmt ein Sprite per INPUT entgegen; es unterscheidet dabei zwischen Leertasten und Punkten einerseits und anderen Buchstaben andererseits, die Tastatur ist auf Dauerfunktion geschaltet.



```
80 REM SPRITE-DREHUNG MIT INPUT
81 :
90 DIMA$(25)
91 POKE53248+32,6:POKE53248+33,6:PRINT"□■"
100 INPUT"ANFANGSWINKEL ";WA:WA=WA*π/180
110 INPUT"▫ENDWINKEL ";WE:WE=WE*π/180
120 INPUT"▫ZAHL DER SCHRITTE ";ZS:S=(WE-WA)/ZS
130 AA=2↑13
135 FOR I=AA TO AA+64*INT((WE-WA)/S+1):POKE I,0:NEXT
138 POKE 650,128
139 PRINT"□    1234567890123456789012345
"
140 FOR I=1 TO 25 :I$=RIGHT$(STR$(I),2)
150 PRINT I$;:INPUT A$(I)
160 NEXT
170 FOR W=WA TO WE STEP S
175 PRINT"▫";W
180 FOR I=1 TO 25
181 A$(I)=A$(I)+"........................."
183 FOR J=1 TO 25
190 Z$=MID$(A$(I),J,1)
200 IF Z$="." OR Z$=" " THEN 250
210 X=INT(11.5+.6*(J-13)*COS(W)-.6*(I-13)*SIN(W))
215 X1=INT(X/8):X2=X-8*X1
220 Y=INT(10+.6*(J-13)*SIN(W)+.6*(I-13)*COS(W))
230 A1=AA+3*Y+X1
240 A2=PEEK(A1):A3=2↑(7-X2):A4=A2 OR A3:POKE A1,A4
250 NEXT:NEXT:AA=AA+64:NEXT
300 V=53248:PRINT"□"
310 POKE 2040,128
320 POKE V+21,1
330 POKE V+39,1
340 POKE V+0,100:POKE V+1,100
350 POKE V+23,PEEK(653):POKE V+29,PEEK(653)
351 TT=9
360 FOR Q=0 TO ZS
370 POKE 2040,128+Q:FOR T=1 TO 2↑(12-TT):NEXT
380 GET TT$:IF VAL(TT$)>0 THEN TT=VAL(TT$)
385 POKE V+23,PEEK(653):POKE V+29,PEEK(653)
390 NEXT
400 GOTO 360
```

DEUS EX MACHINA

Sprites im Programm berechnet

Manchmal soll ein Sprite auch ganz raffiniert aussehen, z.B. wie das Schrägbild einer Schraubenfeder. Das lassen wir am besten den Computer ausrechnen. Das Programm fertigt sich einige Bilder der verschieden stark gedehnten Feder an und baut dann für jeden Zeitpunkt lauter gleiche Bilder (d.h. "verschiedene" Sprites mit gleichem Figurenspeicher) übereinander. Es simuliert damit das Schwingen einer aufgehängten Feder (auch unter dem Einfluß ihrer eigenen Masse). Sie haben sicher Verständnis dafür, daß der C 64 für die Berechnungen etwas Zeit braucht und währenddessen den Fortgang der Prozedur in Zahlen verkündet !

```
1000 REM  WENDELFEDER - EIN TRICKFILM MIT SPRITES
1001 :
1002 REM  NEUSTART MIT RUN 1500
1003 :
1010 FOR K=0 TO 24:PRINT K:S0=8*1024+64*K:L=10+K/2
1011 FOR I=0 TO 62:POKE S0+I,0:NEXT
1015 FOR T=0 TO 2*π STEP.1
1020 X=INT(11+10*COS(T)):X1=INT(X/8):X2=X-8*X1
1030 Y=INT(L/10*T+4*SIN(T))
1040 S1=S0+3*Y+X1:B=PEEK(S1):C=2↑(7-X2):D=B OR C
1050 POKE S1,D
1060 NEXT:NEXT
1500 V=53248:DT=.1:INPUT"D/M       4███";D
1510 INPUT"█GROESSE 1 2     1███";GR
2000 POKE V+21,127 :L=13
2010 FOR J=0 TO 6
2100 POKE V+2*J,100
2120 POKE V+39+J,2
2200 NEXT:J=0
2300 GG=0:IF GR=2 THEN GG=255
2310 POKE V+23,GG:POKE V+29,GG
3000 PRINT"█":FOR J=0 TO 6
3010 POKE 2040+J,128+2*L-20:NEXT:FOR J=0 TO 6
3110 POKE V+1+2*J,50+J*6*L/10*GR
3200 NEXT
3300 L=L+VL*DT:VL=VL-(L-17)*D*DT
3310 GOTO 3000

READY.
```

BÄUMCHEN WECHSLE DICH

Zuordnung der Figuren

In SIMON's BASIC wird das von MOB SET erledigt (unter anderem). Auf jeden Fall muß irgendwo stehen, welche der (vielleicht ganz vielen) Bildfestlegungen als Sprites gerade benutzt werden sollen und mit welchen Nummern das geschehen soll (wichtig für die gegenseitige Priorität: Denken also dagegen an eine Rennbahn mit mehreren Fahrspuren; die Läufer in der Ihnen nächsten Bahn verdecken für Sie manchmal andere). Diese Plätze sind 2Ø4Ø+n, wobei n die Nummer des Sprites (von Ø bis 7) ist. Der Speicherbereich, der als Bild für das jeweilige Sprite verwendet werden soll, wird als Nummer eines 64-Byte-Blocks dort hineingeschrieben. Steht dort also etwa die 128, so heißt das: die Speicherplätze ab 128*64 sollen für dieses Sprite verwendet werden. Tauschen Sie im laufenden Programm durch Anweisungen diese Nummern aus, können die Sprites ihre Gestalt (sprungweise) verändern, z.B. Arme und Beine bewegen, indem die Bildchen einfach ausgetauscht werden. Kleinere Änderungen können Sie aber auch im laufenden Programm im Bit-Muster einer Sprite-Figur machen, z.B. blinkende Augen.

EINSATZKOMMANDOS FÜR SPRITES

Bisher war nur von den Bits die Rede, die die Sprite-Bilder festlegen. Die Beispielprogramme dafür enthalten natürlich auch Anweisungen darüber, was mit ihnen geschehen soll. Das soll nun näher betrachtet werden.

Die Koordinaten

Sprites können im feinen Raster an jede beliebige Stelle des Bildschirms gesetzt werden. Sie dürfen sogar etwas über den Rand hinausragen (wo sie natürlich abgeschnitten zu sehen sind). Damit dabei keine negativen Werte auftreten, sind die

Koordinaten, die wir den Sprites geben müssen, gegen die HIRES-Koordinaten verschoben. Außerdem ist zu beachten, daß bei Multicolor für das Hauptbild (das Wort "Hintergrund" wäre hier mißverständlich) die X-Koordinaten nur halb so große Zahlen sind, während für Sprites immer die gleichen Zahlen zu nehmen sind, unabhängig davon, ob das Sprite selbst oder das Hauptbild Multicolor haben oder nicht.

Füllt ein Sprite seinen Platz voll aus (ist es also 21 Punkte hoch und 24 bzw. 12 Punkte breit), und will man genau bis an die Ränder des Bildes gehen, so sind zu nehmen:

X-Koordinate des Sprites für linke Randlage: 24,
" " " für rechte Randlage 32Ø ohne und
" " " " " " 296 mit
waagerechter Verdoppelung der Sprite-Größe,
Y-Koordinate des Sprites für obere Randlage: 5Ø,
" " " " untere Randlage 229 ohne und
" " " " " " 2Ø8 mit
senkrechter Verdoppelung der Sprite-Größe.

Anders gesagt: Die linke obere Ecke eines Sprites hat für X einen um 24 größeren Wert als der Bildpunkt im HIRES-Format, für Y einen um 5Ø größeren Wert. Damit liegt auch fest, wie weit ein Sprite nach links und nach oben über den Rand laufen darf.

Wo kommen nun diese Koordinaten hin ?

Wenn Sie SIMON's BASIC einsetzen wollen, ist dafür RLOCMOB oder MMOB zuständig (weiter unten !). Die Speicherplätze sind für das Sprite Nr. n für den X-Wert: 53248+2*n, für Y jeweils dahinter, also 53249+2*n. Da Y ohnehin eine Zahl kleiner als 256 ist, ist es kein Problem, sie in einen solchen Speicherplatz zu tun. X kann dagegen auch größer als 255 sein, aber es ist nicht größer als 511. Man zerlegt also X in zwei Teile: X AND 255 sind die niedrigeren 8 Bits, X AND 256 ist das oberste. Nur das Byte aus den unteren 8 Bits kommt in den genannten Speicherplatz. Für die übrigbleibenden Einzelbits gibt es gewissermaßen eine Sammelunterkunft im Speicherplatz 53264: Jedes Bit darin bedeutet

für ein bestimmtes der 8 Sprites, daß sein X-Wert um 256 größer ist als ohne dieses, natürlich der üblichen Numerierung der Bits folgend; steht in 53264 z.B. eine 5, so ist das binär: ØØØØØ1Ø1, und das heißt, daß die Sprites Nr. Ø und Nr. 2 (von rechts nach links von Ø bis 7 !) X-Werte über 255 haben sollen. Das ist ganz schön raffiniert: hier bleibt kein Bit unbenutzt !

Weitere Festlegungen der Sprites

Es ist gar nicht gesagt, daß Sprites überhaupt erwünscht sind, und schon gar nicht, daß es gleich acht Stück sein sollen. Mit den einzelnen Bits in Platz Nr. 53269 werden sie einzeln an- und abgeschaltet. Steht dort z.B. eine 11, also ØØØØ1Ø11, so sind die Geister Nr. Ø, 1 und 3 aktiv.

Wie schon angedeutet, können die Sprites waagerecht und/oder senkrecht auf das doppelte Format aufgeplustert werden. Die Bits in 53271 sind für senkrecht, die in 53277 für waagerecht zuständig.

Die Priorität der Sprites untereinander ist sehr einfach geregelt: Je niedriger ihre Nummer (Ø bis 7), umso weiter vorne, also näher zum Betrachter sind sie, d.h. um so mehr andere Sprites verdecken sie bei Überlappungen. Die Priorität eines jeden Sprites gegenüber dem Hauptbild (d.h. dessen Vordergrundbemalung oder Beschriftung wird in 53275 festgelegt, und zwar gibt dort eine Ø dem Sprite und eine 1 dem Hauptbild den Vorrang.

BUNTE GEISTER

Die Farben der Sprites

Die Farbe für das Sprite Nr. n schreiben Sie in den Platz Nr. 53287+n. Falls es ein MULTIcolor-Sprite sein soll, ist diese Farbe die Nr.2, also das Bitmuster 1Ø in der Bildfestlegung. Nur diese Farbe kann man für alle (bis zu acht)

Sprites getrennt wählen. Die anderen beiden Farben der Multicolor-Sprites sind diesen gemeinsam und kommen in die Plätze 53285 (Nr.1) und 53286 (Nr.3). Wenn das Hauptbild auch Multicolor hat (was nicht unbedingt nötig ist), können dafür andere Farben vorgeschrieben werden; allem gemeinsam ist nur der Hintergrund, den man dort sieht, wo die Sprites entweder gar nicht oder durchsichtig sind und wo zugleich das Hauptbild unbemalt ist. Die Farben des Hauptbildes sitzen übrigens ganz in der Nähe gespeichert:

53280 Umrandung
53281 Hintergrund von allem
53282 Vordergrund des Hauptbildes, Nr. 1 bei Multic.,
53283 Nr. 2 bei Multicolor im Hauptbild
53284 Nr. 3 bei Multicolor im Hauptbild

Zu beachten ist noch, daß in den Plätzen 53285 und 53286, also für die gemeinsamen Sprite-Multicolor-Farben, nur die Farben 0 bis 7 zugelassen sind.

AUF KOLLISIONSKURS

Nicht immer sollen Sprites sang- und klanglos vor einem anderen Gebilde entlangziehen; dazu kann man auf Kollisionen mit dem Hauptbild oder mit anderen Sprites abfragen. Dabei werden nur die undurchsichtigen Teile mitgerechnet. Die Bits in 53279 zeigen an, daß das betreffende Sprite mit einem Vordergrund-element des Hauptbildes zusammengestoßen ist, 53278 entsprechend mit einem anderen Sprite. Diese Speicherplätze können mit PEEK abgefragt werden und müssen natürlich vor der nächsten Abfrage mit POKE wieder auf 0 gesetzt werden. Auf diese Weise kann man Bahnen wirksam begrenzen oder Poltergeräusche aus dem Lautsprecher ertönen lassen, wenn es einen Kontakt gibt.

SIMON-Anweisungen für Sprites

DESIGN m,sp

Ein Speicherbereich, beginnend bei Nr.sp, wird für ein normales (m=Ø) oder Multicolor-Sprite (m=1) reserviert. sp muß dabei ein Vielfaches von 64 sein. Im Falle der hochauflösenden Grafik mit HIRES (mit oder ohne MULTI) ist statt sp sp+48*1Ø24 zu setzen. Es folgen dann 21 Zeilen, die jeweils nach der Zeilennummer einen Klammeraffen haben und dann 24 bzw. 12 Zeichen, die die Farbe der Punkte bedeuten:Punkte für "durchsichtig", B für die Vordergrundfarbe der einfarbigen Sprites, B,C und D für die drei Farben des Multicolor-Sprites.

CMOB b,d ("color movable object block") gibt für Multicolor-Sprites die beiden Farben an, die mit B und D hinter dem Klammeraffen aufgerufen werden; die dritte Farbe (für C) wird in der Anweisung MOB SET mitgeteilt:

MOB SET n,bk,f,p,m   legt einige Dinge für ein Sprite fest:

n die Nummer (Ø bis 7), die zugleich über die Priorität beim Zusammentreffen mehrerer Sprites entscheidet; niedrige Nummer hat Vorrang,

bk Block-Nummer des Speicherbereiches, wo die Gestalt angegeben ist: der Speicherbereich beginnt beim 64-fachen; die in DESIGN verwendete Zahl sp ist also = 64*bk, wenn das Sprite Nr. n auf das in DESIGN festgelegte Bild geschaltet werden soll;

f ist die Vordergrundfarbe des Sprites, im Falle eines Multicolor-Sprites die mit C angewählte Farbe;

p ist die Priorität des (Vordergrundes des) Hauptbildes vor diesem jeweiligen Sprite (1, sonst Ø)

m ist 1 für ein Multicolor-Sprite, sonst Ø

MMOB n,x1,y1,x2,y2,g,t   und   RLOCMOB n,x,y,g,t

bewegen oder setzen ein Sprite auf dem Bildschirm:

n ist die Nummer des Sprites (Ø bis 7),

x,y sind die Koordinaten, genauer: x-24 und y-5Ø sind die HIRES-Koordinaten des linken oberen Eckpunktes des Sprites, x1,y1,x2,y2 sind entsprechend die Koordinatenpaare, zwischen denen das Sprite wandern soll;

g gibt die Größe des Sprites an:

	waag.normal	waag. doppelt
senkr.normal	Ø	1
senkr.doppelt	2	3

t ist die Zeitverzögerung der Wanderung: von 1 (ganz schnell) bis 255 (ganz langsam).

MOB OFF n schaltet das Sprite Nr. n aus.

Kollisionsabfragen:

DETECT Ø bereitet die Abfrage einer gegenseitigen Sprite-Kollision vor, DETECT 1 die nach einer Kollision zwischen einem Sprite und dem Vordergrund des Hauptbildes.
CHECK( ) ist die Abfragefunktion. CHECK(Ø) bekommt den Wert Ø, wenn ein Sprite mit dem Vordergrund des Hauptbildes zusammenstößt, CHECK(n1,n2) wird zu Ø, wenn die Sprites Nr. n1 und Nr. n2 zusammenstoßen.



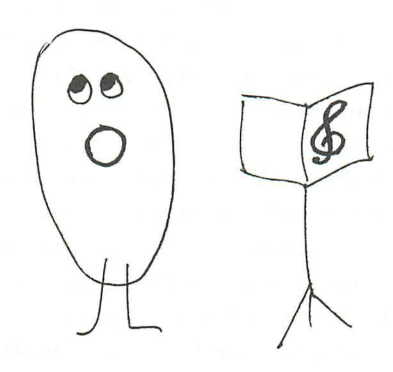
WAS WIRD HIER GESPIELT ?

Grundlagen der Musik und Akustik

Das Wichtigste bei einem Ton ist sicher die Tonhöhe. Akustisch gehört zu ihr die Frequenz. Um zu verstehen, was damit gemeint ist, muß man den Schall etwas genauer betrachten: Unsere Ohren reagieren auf kleine, aber sehr schnelle Luftdruckschwankungen, die sich mehr oder weniger regelmäßig wiederholen. Die Zeit, nach der jeweils (ungefähr) wieder das Gleiche geschieht, heißt Periode oder Schwingungsdauer. Beim Schall sind das Zeiten von rund 1/20 bis 50 Millisekunden. Bildet man die Kehrwerte, so nennt man die Ergebnisse Frequenzen: ein Ton mit der Periode 1/20 Millisekunde hat also die Frequenz 20000/s. Statt 1/s kann man bei der Angabe von Frequenzen auch Hz (Hertz) schreiben. Der Hörbereich eines jungen Menschen (Stunden in Diskotheken zählen dabei wie Wochen oder Monate) geht von etwa 20 bis 20000 Hertz.

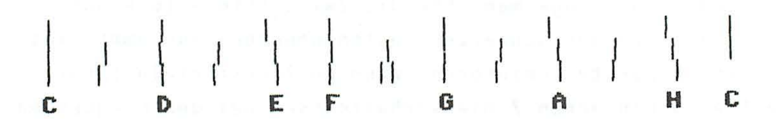
Ein Ton ist um so höher, je größer die Frequenz ist, aber der Zusammenhang ist trotzdem nicht der einfachste, den man sich vorstellen kann: Verdoppelt man die Frequenz mehrmals nacheinander, so empfindet man die Erhöhungen der Tonhöhe wie gleichgroße Stufen; außerdem erscheinen diese Töne einander besonders ähnlich zu sein, in gewisser Weise sogar trotz verschiedener Höhe gleichartig. Bei den Notennamen in

der Musik bekommen sie daher den gleichen Buchstaben (z.B. C, D, E usw.) und werden nur durch kleine Ziffern oder andere Zeichen (Striche) unterschieden. Diesen speziellen Tonhöhensprung (Intervall) nennt man eine Oktave (Grund: Bei der Tonleiter ist der 8. Ton die Oktave zum 1., die Bezeichnung ist genau so unlogisch wie die Redensart "8 Tage", wenn man eine Woche meint; man sollte besser nicht Anfang und Ende mitzählen). Eine Oktave bedeutet also eine Verdoppelung der Frequenz. Was ist nun aber mit den Schritten dazwischen ?



Die Dur-Tonleiter (durus = hart) legt 5 größere und zwei kleinere Intervalle in die Oktave, und zwar nach diesem Muster: I II III IV V VI VII VIII
Die großen heißen Ganztonschritte, die kleinen Halbtonschritte, eine andere Bezeichnung ist: große und kleine Sekunden. So lange man ohne die Zwischentöne (die auf der Klaviatur zu den schwarzen Tasten gehören) auskommt, ist es am saubersten anzuhören, wenn möglichst viele Intervalle zu einfachen Zahlenverhältnissen bei den Frequenzen gehören. Bei Saiteninstrumenten sind das zugleich die Verhältnisse, in denen man Längen auf der gleichen Saite abgreifen muß, um die entsprechenden Töne zu zupfen oder zu streichen (genauer: die Frequenzen verhalten sich bei einer Saite umgekehrt wie die abgegriffenen Längen, wenn sonst nichts verändert wird). Schon die alten Griechen (besonders die Schule des Pythagoras) wußten, daß schöne

Musik etwas mit einfachen Zahlen zu tun hat. Eine Folge von Zahlen mit einfachen Verhältnissen, die sich als Dur-Tonleiter eignet, ist: 24 27 30 32 36 40 45 48. Insbesondere ist die Quinte (das Intervall von I nach V) mit dem Zahlenverhältnis 3:2 verbunden. Andererseits entsprechen einer Quinte 7 Halbtonschritte. Zwei Quinten übereinander sind also 14 Halbtöne, deren 12 sind eine Oktave; es bleibt also noch ein Ganzton übrig. Man könnte nun versuchen, alle 12 Töne innerhalb einer Oktave aus Quinten, also mit dem Frequenzverhältnis 3:2 zu erzeugen. Leider geht die Rechnung nicht auf, denn 2^7 ist nicht genau das gleiche wie $(1,5)^{12}$. Die Folge ist: Bei einer solchen reinen Quintenstimmung bekommt man für die Töne zwischen denen der Durtonleiter (also für die schwarzen Tasten) verschiedene Frequenzen, je nachdem ob man sie als erniedrigte oder als erhöhte Töne auffaßt. Wenn man also auf weißen und schwarzen Tasten spielen will und dabei die schwarzen nicht nur sporadisch benutzen will, gibt es mit den bisher beschriebenen Systemen Probleme. Im 18. Jahrhundert hat sich darum für Tasteninstrumente ein anderes System durchgesetzt: die "wohltemperierte Stimmung" mit 12 genau gleichen Intervallen, also stets dem gleichen Frequenzverhältnis zwischen zwei benachbarten (Halb-)Tönen. Da nun 12 solche Halbtonschritte eine Oktave, also den Frequenzfaktor 2, bilden sollen, muß das Frequenzverhältnis für den Halbton (= kleine Sekunde) die 12. Wurzel aus 2 sein, also in BASIC: 2↑(1/12), es ist 1.05946309.



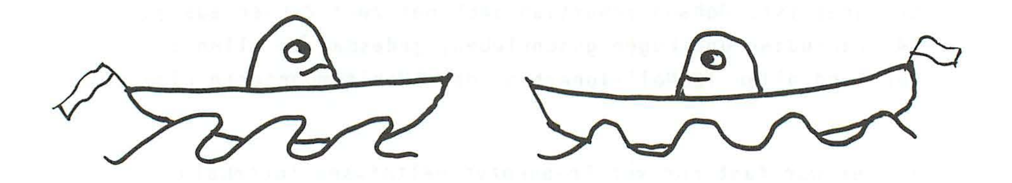
Die Abbildung stellt auf einer logarithmischen Frequenzskala (also einer gleichmäßigen Tonhöhenskala !) die drei beschriebenen Möglichkeiten dar: oben die "reine Durtonleiter" mit den Verhältnissen 24:27:30 usw. , darunter die gleichmäßige (wohltemperierte) Stimmung, als drittes die

reine Quintenstimmung C Des D Es F Ges/Fis G As A B H C. Theoretisch sollten Fis und Ges gleich sein, bei der reinen Quintenstimmung (von C ausgehend) liegt aber Fis höher als Ges. Diese Diskrepanz ist als "pythagoreisches Komma" in der Musiktheorie bekannt. Zum Programmieren empfiehlt sich meist die wohltemperierte Stimmung, da sie mathematisch am einfachsten ist (die gleiche Formel für alle Töne, keine DATA nötig), und da sie für alle Töne, auch die Halbtöne geeignet ist. Johann Sebastian Bach hat zwei Zyklen aus je 24 Praeludien und Fugen geschrieben, jedesmal in allen 12 Dur- und allen 12 Moll-Tonarten: das "Wohltemperierte Clavier".

Bisher war fast nur von Frequenzverhältnissen innerhalb der Oktave die Rede. Welche Frequenzen haben nun die Töne, die auf den gemeinsamen Namen C hören ? Die Physiker sagen: am besten glatte Potenzen von 2, also 32 Hz, 64 Hz usw. Die Musiker haben früher gesagt: der Kammerton A soll 435 Hz haben; später haben sie auf 440 Hz erhöht, und bei elektronischen Orgeln findet man heute 442 Hz eingestellt. Die Unterschiede sind nicht sehr groß: Sie haben die Auswahl. Um nun die Frequenz eines beliebigen Tones berechnen zu lassen, sagen Sie dem Computer für einen bestimmten Ton die Frequenz (z.B. die 32 Hz für ein sehr tiefes C oder die 440 Hz für ein A) und geben für die übrigen Töne an, wieviele Halbtöne er darüber liegt. K Halbtöne über 32 Hz liegt dann die Frequenz 32*2↑(K/12) (vgl. Zeile 511Ø im SYNTHESIZER-Programm).

Nun hat der C 64 für die Frequenz einer jeden seiner drei Stimmen 2 Bytes, das gibt also jeweils 2^{16}= 65536 mögliche Werte. Anhang P des Benutzer-Handbuchs enthält lange Tabellen, aus denen nur der Faktor 17.Ø295 zwischen dem dort als Parameter bezeichneten Wert und der Maßzahl der Frequenz wichtig ist; alles andere erledigen wir durch Rechnungen, die der Computer im Vorlaufteil des Programms ausführen darf. Die 16-Bit-Zahlen (nämlich diese Parameter) müssen in zwei 8-Bit-Zahlen aufgespalten werden. Im Dezimalsystem entspricht das der Aufteilung einer 6-stelligen

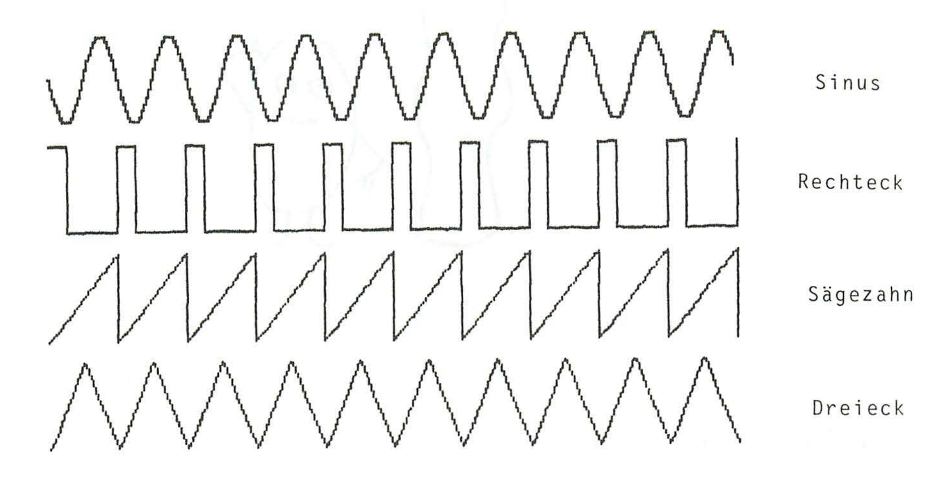
Zahl in zwei 3-stellige o.ä. Das "obere" Byte erhalten wir durch Division durch 256 (denn das ist 2^8) und Abschneiden mit INT, das untere Byte ist einfach der Rest, also FQ-256* INT(FQ/256). Diese beiden Bytes müssen in die Frequenzregister, die für die Stimme Nr. Y die Speicherplätze 54272+Y*7 für das untere und 54273+Y*7 für das obere haben.



DIVERSE ZACKEN

Schwingungsformen

Nun weiß der Rechner also, welche Frequenz(-en) er spielen soll. Das Schöne an einem Synthesizer ist hauptsächlich, daß er so viele Instrumente nachmachen und noch mehr andere sozusagen künstlich bilden kann. Worin unterscheiden sich denn verschiedene Instrumente, die den gleichen Ton spielen ? Schließt man ein Mikrofon an ein Oszilloskop, so findet man unterschiedliche Schwingungsformen, die bei waagerechter Zeitablenkung wie Wellen mit oder ohne Kräuselung oder wie Sägezähne usw. aussehen. Man spricht daher meist von Wellenform, obwohl nicht der physikalische Begriff der Welle (bei der es auf Abhängigkeiten von Ort und Zeit ankommt), sondern der der Schwingung (nur Zeitabhängigkeit von Bedeutung) betroffen ist. Akustisch und in gewisser Weise auch mathematisch am einfachsten ist eine sinusförmige Schwingung; man kann sie am einfachsten bei einer Stimmgabel hören. Der C 64 benutzt jedoch andere Schwingungsformen, die bei der digitalen Erzeugung einfacher sind: das Rechteck, das (symmetrische) Dreieck und den Sägezahn (das asymmetrische Dreieck mit einer

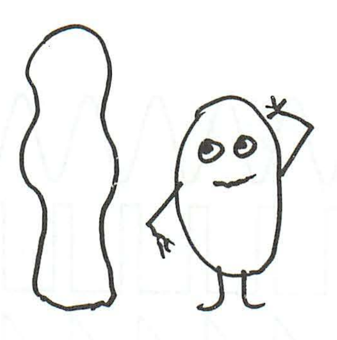


senkrechten Flanke). Die Darstellung bedeutet dabei die Auftragung z.B. der elektrischen Spannung gegen die Zeit. Beim Rechteck kann noch das Verhältnis der Zeitabschnitte mit hoher und mit niedriger Spannung verändert werden, und zwar in 2^{16} Schritten. Dieses Tastverhältnis bestimmt also (bei gegebener Frequenz !) die Pulsbreiten. Die beiden Register nach den Frequenzregistern sind die richtigen Adressen für diese Information: 54274+7*Y für das untere Byte, 54275+7*Y für das obere (Y die Nummer der Stimme von Ø bis 2, Zerlegung wie bei der Frequenz besprochen).

Welche der Schwingungsformen verwendet werden soll, wird pro Stimme mit einem einzelnen Bit festgelegt; in den Registern 54276+Y*7 schaltet je ein Bit eine Schwingungsform ein:

2↑7= 128 Rauschen
2↑6= 64 Rechteck
2↑5= 32 Sägezahn (asymm. Dreieck)
2↑4= 16 (symmetrisches) Dreieck

Es ist dabei im allgemeinen zu empfehlen, nur eins dieser vier Bits auf 1 zu setzen.



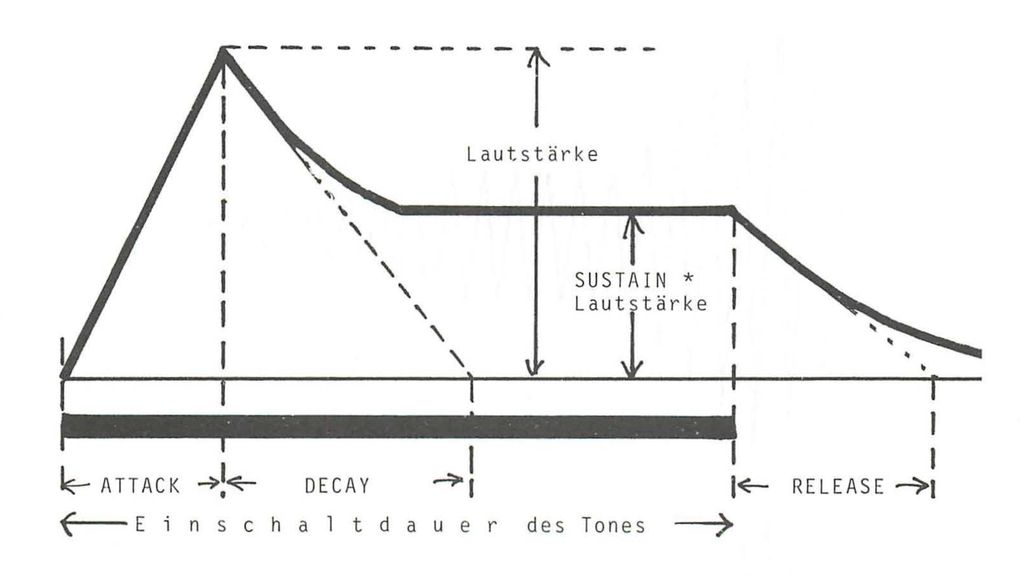
HÜLLKURVEN

Töne verschiedener Instrumente unterscheiden sich nicht nur in der Zusammensetzung des Klanges aus Grund- und Obertönen, sondern auch im Ein- und Ausschwingverhalten: Sie beginnen nicht gleich in voller Stärke, und oft klingen sie deutlich nach. Der Synthesizer erfaßt dieses Verhalten durch eine Hüllkurve, auch ADSR-Kurve genannt, weil sie durch 4 Angaben beschrieben wird:

A = Attack = Anstiegszeit (von Ø auf gewählte Lautstärke)
D = Decay = Abklingzeit (von voller Stärke auf Ø, wird aber beim Erreichen der Sustain-Lautstärke nicht fortgesetzt)
S = Sustain = ausgehaltene Lautstärke (als Bruchteil der vollen Lautstärke)
R = Release = Nachklingen nach Abschalten des Tones

Der Ton wird mit dem Ø-Bit der Wellenform ein- und ausgeschaltet. Beim Einschalten beginnt die Lautstärke bei 0 und steigt linear auf den eingestellten Wert an. Die dafür benötigte Zeit liegt zwischen 2 Millisekunden (bei ATTACK Ø) und 8 Sekunden (bei ATTACK 15); die Schritte dazwischen sind jeweils Faktoren zwischen 1,2 und 3. Sogleich fällt die Lautstärke wieder, und zwar um so schneller, je kleiner DECAY eingestellt ist (haben ATTACK und DECAY den gleichen Wert, so wird auch 0 wieder nach der

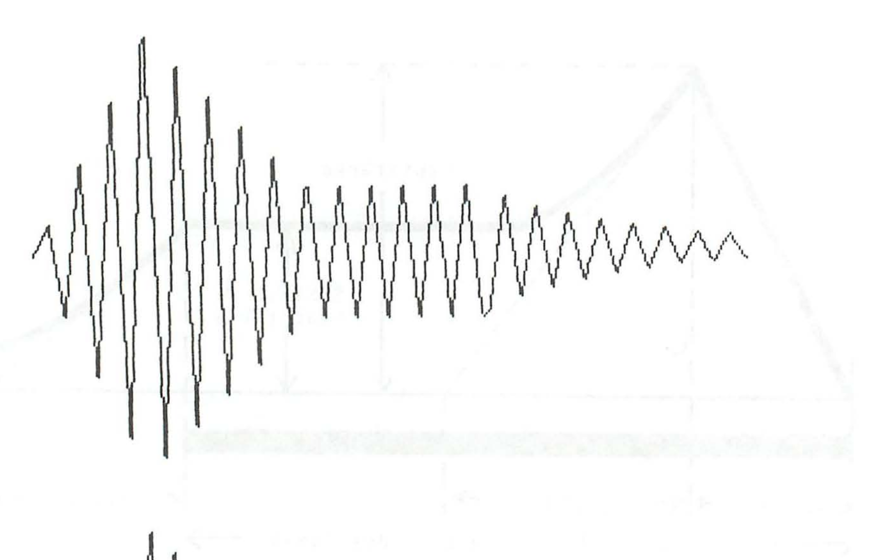
gleichen Zeit erreicht, falls nicht das SUSTAIN-Niveau vorher erreicht wird). SUSTAIN gibt keine Zeit, sondern eine relative Lautstärke an: 15 ist das (als allgemeine Lautstärke festgelegte) Maximum, 8 etwa die Hälfte davon, 0 natürlich 0. Für den Rest der Einschaltdauer bleibt der Ton nun auf dieser Lautstärke. Durch Abschalten (d.h. Nullsetzen des Ø-Bits in der Wellenform) geht man zum Nachklang (RELEASE) über, falls sonst nichts abgeschaltet wurde (also auch nicht die übrigen Bits der Wellenform). Das geschieht auch, wenn DECAY oder gar ATTACK noch gar nicht beendet worden sind: die Lautstärke klingt nun exponentiell ab, bis ein neuer Ton in der selben Stimme eingeschaltet wird.



Die ADSR-Kurve wird in vielen Büchern über den C 64 irreführend beschrieben. Die 4 Zahlen für ATTACK, DECAY, SUSTAIN und RELEASE beziehen sich nämlich nicht durchweg auf die Dauer dieser Abschnitte, sondern indirekt auf die Steigungen und die Werte der Lautstärke. SUSTAIN ist dabei als Faktor aufzufassen, der mit 1/15 der allgemeinen Lautstärke malgenommen wird. Die anderen drei Werte bestimmen An- und

Abstiegszeiten, aber sie sind nicht proportional zu ihnen, sondern es bedeutet ein Schritt jeweils eine Erhöhung um ungefähr den Faktor 1,7, von wenigen Millisekunden (bei Ø) bis zu mehreren Sekunden (bei 15).

Die folgenden Bilder zeigen, wie im Falle niedriger Frequenzen die Schwingungen in ihrer Amplitude von der ADSR-Kurve bestimmt werden; für hohe Frequenzen wären die Linien so dicht, daß sie nicht mehr zu trennen wären.



Ein Demo-SYNTHESIZER

Im Gegensatz zu dem sonstigen Stil dieses Buches sollen die Musikfunktionen des C 64 nicht anhand kurzer Einzelprogramme, sondern an einem etwas größeren Demo-Programm diskutiert werden. Der Grund liegt darin, daß auch für einfache Effekte jeweils eine ganze Reihe von Speicherplätzen sinnvoll belegt werden müssen. Das hier abgedruckte Synthesizer-Programm hat folgende Eigenschaften:

es belegt alle zur Musik gehörenden internen Speicher im direkten Zugriff über die Tastatur (wobei lediglich einige fein einstellbaren Werte nur in groben Schritten verstellt werden),

es zeigt diese Werte auf dem Bildschirm vollständig an

es ordnet zwei Reihen der Tastatur den weißen und schwarzen Tasten einer Klaviatur zu; auf ihnen kann man einstimmig in Echtzeit spielen,

gleichzeitig kann man mit der linken Hand die drei (i.a. verschieden registrierten) Stimmen in beliebigen Kombinationen koppeln (3 Einzelstimmen, 3 paarweise Kopplungen und "volles Werk"), indem man gleichzeitig eine bis drei der Umschalttasten SHIFT (auch einrastbar), C= und CTRL drückt.

Der Synthesizer erscheint in diesem Buch in zwei Versionen, zuerst mit Benutzung der HIRES-Grafik, was besonders die ADSR-Kurven sehr anschaulich macht. Falls Sie aber SIMON's BASIC nicht haben, können Sie auf die zweite Version ausweichen, in der ebenfalls alle Informationen ins Bild gesetzt

werden, allerdings nur mit Balkengrafik. Beide Versionen sind auch bei Hagemann auf einer Cassette erhältlich.Wenn Sie eines der beiden Programme selbst eintippen, denken Sie bitte an die Editiertricks, mit denen Sie aus jedem Text auf dem Bildschirm neue Programmzeilen machen können, also insbesondere eine Serie von ähnlichen Zeilen relativ bequem schreiben können. Wenn Sie mit dem SID näher vertraut sind, können Sie das Programm auch zum Ausgangspunkt eigener Varianten machen: z.B. Verzicht auf Stimmenwahl über Tasten und stattdessen vorwählbare Register oder eingebaute Melodien usw.Natürlich können Sie auch kleine Programme mit Einzeleffekten, auch als Unterprogramme für Spiele etc. schreiben.

Wenn nach dem Eingeben und Starten des Programmes dieses Bild fertig ist, können Sie musizieren (wobei Sie stets mindestens eine der drei Tasten SHIFT, C= und CTRL gedrückt halten müssen):

SHIFT | C= | CTRL

A D S R
X U N
Z C B M

RA A
RE S
SA D
DR F
TB G
RM H
S,Y J
PB ←1
OK .
FI :

3 f1
HP f3
BP f5
TP f7

FF :=
RS KL
LS

Klaviatur:
23 567 90 -£
QWERTYUIOP@*↑

```
700 REM  SYNTHESIZER MIT HIRES-GRAFIK
701 :
800 HIRES 7,0:PE=104:SI=54272
801 LINE 0,164,1023,164,1
802 LINE 105,0,105,164,1
803 LINE 210,0,210,164,1
809 TEXT0,0,"  SHIFT      C=       CTRL",1,1,12
810 FOR Y=0 TO 2:FOR K=0 TO 15:LINE 10+Y*PE+K*6,35,10+Y*PE+K*6,38,1:NEXT:NEXT
821 FOR Y=0 TO 2:FOR K=0 TO 1:LINE 40+Y*PE+K*30,35,40+Y*PE+K*30,42,1:NEXT:NEXT
822 TEXT 5,48,"   A D S R      A D S R      A D S R ",1,1,8
823 TEXT 5,56,"+  X V N ,   +  X V N ,   +  X V N , +",1,1,8
824 TEXT 5,64,"-  Z C B M   -  Z C B M   -  Z C B M -",1,1,8
825 DATA"RA A","RE S","SA D","DR F","TB G","RM H","SY J","PB ←1","OK ./","FI :"
826 FOR I=0 TO 9:READ TE$:TEXT 0,72+9*I,TE$,1,1,8:NEXT
830 DATA"▒  F1","▒|┐ F3","▒|┐ F5","▒|┐ F7"
831 FOR I=0 TO 3:READ T$:TEXT 0,168+8*I,T$,1,1,8:NEXT
900 TEXT 220,170,"▒LAVIATUR:",1,1,8
901 TEXT 220,180,"23 567 90 -£",1,1,8
902 TEXT 215,190,"QWERTYUIOP@*↑",1,1,8
1010 GOTO 5000
1150 POKE SI+RR,R(RR):RETURN
1200 X=PEEK(203):YY=PEEK(653)
1210 IF X=64 THEN 1300
1220 IF YY=0 THEN 1200
1230 IF AD(X)=1 THEN 1500
1240 FOR I=0 TO 2:IF (YY AND 2↑I) THEN Y=I
1250 NEXT
1260 ON AD(X)GOTO1500,2200,2300,2400,2500,2600,2700,2800,2900,3000,3100,3200,330
0
1300 FOR Y=0 TO 2:POKE SI+4+Y*7,R(4+Y*7):NEXT:GOTO 1200
1500 FOR Y=0 TO 2
1510 IF (YY AND 2↑Y) =0 THEN 1560
1520 POKE SI  +7*Y,F2%(12*OK(Y)+AR(X))
1530 POKE SI+1+7*Y,F1%(12*OK(Y)+AR(X))
1540 POKE SI+4+7*Y,R(4+7*Y) OR 1
1560 NEXT:GOTO 1200
```

```
2200 RR=4+7*Y:IF (R(RR) AND 2↑AR(X)) THEN R(RR)=R(RR)-2↑AR(X):GOSUB1150:GOTO 223
0
2210 RR=4+7*Y:IF (R(RR) AND 2↑AR(X))=0 THEN R(RR)=R(RR)+2↑AR(X):GOSUB1150:GOTO 2
240
2230 BLOCK 50+PE*Y,135-AR(X)*9,98+PE*Y,142-AR(X)*9,0:GOTO 1200
2240 BLOCK 50+PE*Y,135-AR(X)*9,98+PE*Y,142-AR(X)*9,1
2241 TEXT 50+PE*Y,135-AR(X)*9,W$(AR(X)),0,1,8:GOTO 1200
2300 RR=5+7*Y:Z=AK(Y):GOSUB7000:AK(Y)=Z:R(RR)=16*AK(Y)+DE(Y):G=AK(Y):ZE=1:GOTO 2
650
2400 RR=5+7*Y:Z=DE(Y):GOSUB 7000:DE(Y)=Z:R(RR)=16*AK(Y)+DE(Y):G=DE(Y):ZE=2:GOTO
2650
2500 RR=6+7*Y:Z=SU(Y):GOSUB 7000:SU(Y)=Z:R(RR)=16*SU(Y)+RE(Y):G=SU(Y):ZE=3:GOTO
2650
2600 RR=6+7*Y:Z=RE(Y):GOSUB 7000:RE(Y)=Z:R(RR)=16*SU(Y)+RE(Y):G=RE(Y):ZE=4
2650 GOSUB 1150:GOSUB 8000:GOTO 1200
2700 OK(Y)=OK(Y)+AR(X)
2710 IF OK(Y)<0 THEN OK(Y)=0
2720 IF OK(Y)>5 THEN OK(Y)=5
2730 BLOCK 50+PE*Y,144,90+PE*Y,151,0
2731 TEXT 50+PE*Y,144,O$(OK(Y)),1,1,8:GOTO 1200
2800 Z=VO:GOSUB 7000:VO=Z:R(24)=(R(24) AND 240)+VO:RR=24:GOSUB 1150
2810 BLOCK 150,192,195,198,0:BLOCK 150,192,150+VO*3,198,1:GOTO 1200
2900 Z=RS:GOSUB 7000:RS=Z:R(23)=16*RS+FI:ZE=19:RR=23:GOSUB 1150
2910 BLOCK 150,180,195,186,0
2911 BLOCK 150,180,150+3*INT(R(23)/16),187,1:GOTO 1200
3000 Z=RF:GOSUB 7000:RF=Z:R(22)=16*RF:ZE=18:RR=22:GOSUB 1150
3010 BLOCK 150,168,195,174,0
3011 BLOCK 150,168,150+3*INT(R(22)/16),175,1:GOTO 1200
3100 RR=3+7*Y:Z=PU(Y):GOSUB 7000:PU(Y)=Z:R(RR)=PU(Y):ZE=13:GOSUB 1150
3110 BLOCK 50+PE*Y,135,95+PE*Y,141,0
3111 BLOCK 50+PE*Y,135,50+PE*Y+PU(Y)*3,141,1:GOTO 1200
```

```
3200 RR=23:IF (FI AND 2↑Y)=2↑Y THEN FI=FI-2↑Y:GOTO 3220
3210 IF (FI AND 2↑Y)=0   THEN FI=FI+2↑Y:GOTO 3230
3220 R(23)=FI+16*RS:BLOCK 50+PE*Y,153,98+PE*Y,161,0:GOSUB 1150:GOTO1200
3230 R(23)=FI+16*RS:BLOCK 50+PE*Y,153,98+PE*Y,161,1
3231 TEXT 50+PE*Y,154,"FILTER",0,1,8:GOSUB 1150:GOTO 1200
3300 RR=24:IF (R(24) AND 2↑AR(X)) THEN R(24)=R(24)-2↑AR(X):GOTO 3350
3310 IF (R(24) AND 2↑AR(X))=0 THEN R(24)=R(24)+2↑AR(X):GOTO 3360
3350 BLOCK 50,224-AR(X)*8,90,231-AR(X)*8,0:GOSUB 1150:GOTO 1200
3360 BLOCK 50,224-AR(X)*8,90,231-AR(X)*8,0:GOSUB 1150
3361 TEXT 50,224-AR(X)*8,GF$(AR(X)),1,1,8:GOTO 1200
4000 FOR I=0 TO 24:POKE SI+I,R(I):NEXT
4100 FOR Y=0 TO 2:AK(Y)=R(5+7*Y)/16:DE(Y)=(R(5+7*Y)AND15)
4110 SU(Y)=R(6+7*Y)/16:DE(Y)=(R(6+7*Y)AND15):GOSUB 8000:NEXT
4200 FOR Y=0 TO 2:FOR X=1 TO 7
4201 IF (R(4+7*Y)AND 2↑X)=0 THEN 4230
4210 BLOCK 50+PE*Y,135-X*9,98+PE*Y,142-X*9,1
4220 TEXT 50+PE*Y,135-X*9,W$(X),0,1,8
4230 NEXT:NEXT
4300 FOR Y=0 TO 2:BLOCK 50+PE*Y,135,50+PU(Y)*3,141,1
4301 LINE 50+PE*Y,142,95+PE*Y,142,1:NEXT
4305 FOR Y=0 TO 2:TEXT 50+PE*Y,144,O$(OK(Y)),1,1,8:NEXT
4310 FOR Y=0 TO 2:IF (FI AND 2↑Y)=0 THEN 4330
4320 BLOCK 50+PE*Y,153,98+PE*Y,161,1
4321 TEXT 50+PE*Y,154,"FILTER",0,1,8
4330 NEXT
4410 IF (R(24) AND 64) THEN TEXT 50,176,"__▬",1,1,8
4420 IF (R(24) AND 32) THEN TEXT 50,184,"_▬_",1,1,8
4430 IF (R(24) AND 16) THEN TEXT 50,192,"▬__",1,1,8
4500 TEXT 100,168,"FF ;=",1,1,8:BLOCK 150,168,150+3*INT(R(22)/16),175,1
4501 LINE 150,175,195,175,1
4510 TEXT 100,180,"RS KL",1,1,8:BLOCK 150,180,150+3*INT(R(23)/16),187,1
4511 LINE 150,187,195,187,1
4520 TEXT 100,192,"LS ▒Q]",1,1,8:BLOCK 150,192,150+VO*3,198,1
4600 GOTO 1200
```

```
5000 DATA 0,0,128,1,64,0,128,0,0,128,1,32,34,162,0,0,128,1,16,68,200,0,128,41,79
5010 DIM R(24):FOR I=0 TO 24:READ R(I):NEXT
5020 FOR I=0 TO 2:OK(I)=I+2:AK(I)=I*2:DE(I)=I*2:SU(I)=I*2+8:RE(I)=I*2:NEXT:VO=15
5100 DIM F1%(84),F2%(84):FOR I=0 TO 5:FOR J=0 TO 11:K=12*I+J
5110 FR=32*2↑(K/12):FQ=FR*17.0295
5120 F1%(K)=INT(FQ/256):F2%(K)=FQ-F1%(K)*256:NEXT:NEXT
5200 DATA 0,0,0,0,8,1,13,4,13,7,13,6,13,5,8,-1,1,3,1,2
5201 DATA 2,7,0,0,3,-1,2,6,1,4,0,0,1,6,1,5,2,5,1,8
5202 DATA 4,-1,2,4,1,7,3,1,1,10,1,9,2,3,0,0,5,-1,2,2
5203 DATA 1,11,4,1,1,13,1,12,2,1,1,15,6,-1,9,-1,1,14,5,1
5204 DATA 0,0,1,16,9,1,1,18,7,-1,12,0,1,17,6,1,1,20,1,19
5205 DATA 10,-1,1,22,0,0,10,1,1,21,7,1,11,1,11,-1,0,0,1,1,0,0,0,0,1,0
5210 DIM AD(62),AR(62):FOR I=0 TO 62:READ AD(I),AR(I):NEXT:FI=1
5340 DIM W$(7):FOR I=0 TO 7:READ W$(I):NEXT
5350 DATA"      "," SYNC "," RING "," TEST ","","","",""
5360 DATA "      ","      ","      ","      ","      ","      "
5361 FOR I=0 TO 5:READ O$(I):NEXT
5370 DATA"","","","STUMM"
5371 FOR I=4 TO 7:READ GF$(I):NEXT
5500 GOTO 4000
7000 Z=Z+AR(X)
7010 IF Z<0  THEN Z=0
7020 IF Z>15 THEN Z=15
7030 RETURN
8000 BLOCK 5+Y*PE,15,100+Y*PE,30,0
8020 LINE 40+Y*PE-2*AK(Y),30,40+Y*PE,15,1
8021 LINE 40+Y*PE,15,40+Y*PE+2*DE(Y),30,1
8022 LINE 40+Y*PE+2*DE(Y),30,40+Y*PE-2*AK(Y),30,1
8023 IF AK(Y)>1 THEN PAINT 39+Y*PE,29,1
8024 IF DE(Y)>1 THEN PAINT 41+Y*PE,29,1
8030 BLOCK 40+Y*PE,30-SU(Y),70+Y*PE,30,1
8031 LINE 70+Y*PE,30-SU(Y),70+Y*PE+2*RE(Y),30,1
8032 LINE 70+Y*PE+2*RE(Y),30,70+Y*PE,30,1
8033 IF SU(Y)>1 THEN IF RE(Y)>1 THEN PAINT 71+Y*PE,29,1
8050 RETURN
```

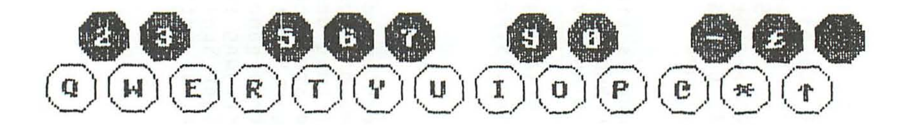
Zeilenkommentar zum Programm "SYNTHESIZER MIT HIRES-GRAFIK"

8ØØ	PE ist die Spaltenbreite in der Grafik, SI die Basisadresse des SID
8Ø1-9Ø2	Startzustand der Grafik 81Ø-821 Skalenstriche für ADSR
115Ø	Unterpr. zum Einlesen einer Änderung in das SID
12ØØ	Ablesen der aktuellen Tastendrücke: 2Ø3 erfaßt eine "normale" Taste, 653 die Kombination von SHIFT, C= und CTRL
121Ø	keine "normale" Taste: Abschalten des Tones
122Ø	keine Stimme gedrückt
123Ø	eine Melodietaste gedrückt
124Ø-125Ø	falls der Klang geändert werden soll und es sind mehrere Stimmen gedrückt, wird die mit der niedrigsten Nummer genommen
126Ø	Sprungverteiler für die verschiedenen Klangänderungen
13ØØ	Unterpr. zum Abschalten eines Tones
15ØØ-156Ø	Setzen der Frequenz mit Beachtung der Oktavlage
22ØØ-2241	Schwingungsform, Testbit, Ringmodulation, Synchronisation
23ØØ-265Ø	ADSR-Kurve
27ØØ-2731	Oktave
28ØØ-281Ø	gemeinsame Lautstärke
29ØØ-2911	Resonanzstärke des Filters
3ØØØ-3Ø11	Resonanzfrequenz des Filters
31ØØ-3111	Pulsbreite, d.h. Tastverhältnis für Rechteckschwingung
32ØØ-3231	das Filter wird für die gewählte Stimme ein- oder ausgeschaltet
33ØØ-3361	Hochpaß-, Bandpaß- und/oder Tiefpaß-Wirkung des Filters werden ein- oder ausgeschaltet, Stimme wird stumm oder hörbar gesetzt
4ØØØ-46ØØ	Setzen des SID und des Grafikbildes beim Start des Programms
5ØØØ-5Ø2Ø	Startwerte für das SID (Sie können hier eigene Werte setzen, die dann nach dem Start sogleich zur Verfügung stehen)
51ØØ-512Ø	Berechnung der Frequenzen und zugehörigen Speicherinhalte
52ØØ-521Ø	Steuerzahlen, die den einzelnen Tasten die entsprechenden Teile des Programms und Zahlen zuweisen
534Ø-5371	Zeichen für die Grafik
7ØØØ-7Ø3Ø	Unterpr. für die Änderung eines Wertes Z unter Begrenzung auf den Bereich von Ø bis 15
8ØØØ-8Ø5Ø	Unterpr. zum Zeichnen einer ADSR-Kurve

Wie spielt man mit dem Programm ?

Prinzipiell müssen Sie stets eine der drei Umschalttasten drücken, wenn Sie etwas hören wollen (dann dürfen Sie sogar zwei oder alle drei zugleich drücken), aber auch wenn Sie registrieren, also den Klang verändern (dann sinnvollerweise nur eine). Das gilt auch für die Änderungen, die für alle Stimmen gemeinsam gelten. Für die Stimme 1 können Sie auch einfach die SHIFTLOCK-Taste einrasten lassen, allerdings hört der letzte Ton dann nie auf.

Die Klaviertasten beginnen bei Q mit der Note C:

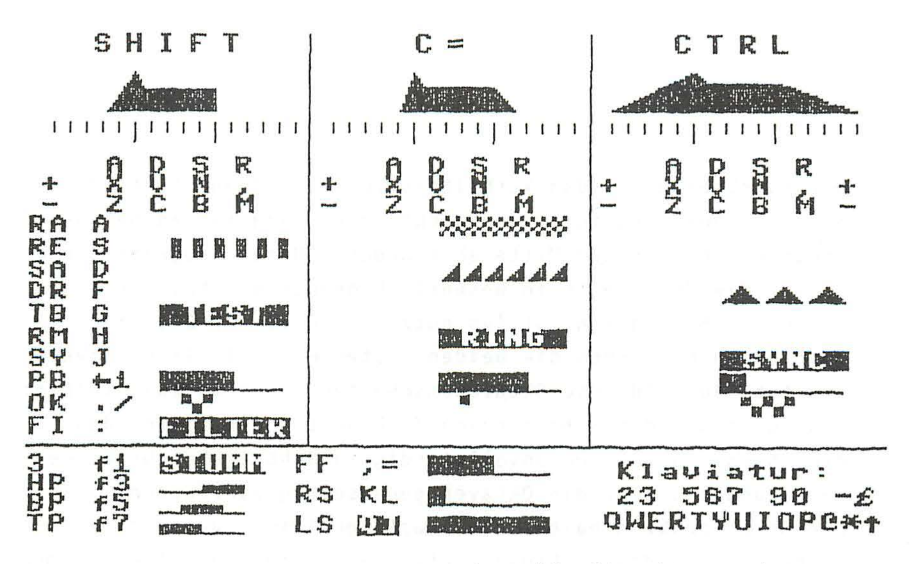


Am besten legen Sie sich oberhalb der Tastatur auf die Konsole eine Schablone, die die unbesetzten Tasten zur besseren Orientierung abdeckt. Beachten Sie, daß der C 64 stets nur eine Taste zur gleichen Zeit zur Kenntnis nimmt (ausgenommen sind davon nur die drei Umschalttasten). Sie können also mit dem Programm in der vorliegenden Form nur "unisono" spielen, also können gleichzeitig nur Oktaven oder Primen erklingen.
Um den Klang umzuschalten, drücken Sie zugleich die entsprechende Taste für die Funktion und die für die Stimme: um das Ausklingen (RELEASE) von Stimme 3 zu verkürzen, müssen Sie also CTRL und das Komma drücken. Warten Sie dann, bis die Änderung auf dem Bildschirm angezeigt wird, bevor Sie weiterspielen oder weitere Änderungen vornehmen. Auf die Hüllkurve kommen wir später noch zu sprechen, auch auf die Wellenformen, über die hier schon gesagt werden muß, daß Sie am besten immer nur eine einstellen pro Stimme.

Kommentar zum Display des Synthesizer-Programms

Das Bild enthält eine geballte Ladung von Information, nämlich in grafischer Form alle für die Musik bedeutsamen Speicherinhalte, mit Ausnahme derer, die man beim Spielen einer Melodie unmittelbar verändert: Frequenzen, Stimme(n) und Ein/Aus. Der Bildschirm ist in drei große Teile für die drei Stimmen und einen gemeinsamen Streifen ganz unten aufgeteilt. Die obersten Zeilen geben an, welche Umschalttaste den Stimmen zugeordnet ist: sie muß stets gedrückt sein, wenn die entsprechende Stimme erklingen soll, oder wenn für diese Stimme etwas umgeschaltet werden soll. Aber auch wenn allgemeine Dinge umgeschaltet werden, muß eine der drei Tasten gedrückt sein (welche, ist dann gleichgültig).

Die Bilder darunter geben die ADSR-Kurve schematisiert an: In Wirklichkeit verändern sich die Längen auf der Zeitskala nicht von Ø bis 15, sondern um mehr als den Faktor 4000, was nicht mehr deutlich abgebildet werden kann. Die Striche



Keine sinnvolle Eingabe, sondern nur zur Erklärung des Grafik-Displays !

unter den Kurven erleichtern das Wiederfinden bestimmter Einstellungen. In den Zeilen mit + und - sind die Tasten angegeben, die man (zusammen mit SHIFT, C= oder CTRL) drükken muß, um A, D, S bzw. R zu vergrößern bzw. verkleinern (Vergrößern heißt bei Attack, Decay und Release Verlängern, bei Sustain Erhöhen der Lautstärke). Diese acht Tasten befinden sich in der untersten Reihe auf der Tastatur.

Die nächsten 4 Zeilen betreffen die Schwingungsformen (meist Wellenformen genannt): Rauschen, Rechteck, Sägezahn (d.h. Dreieck mit einer ganz steilen Flanke) und Dreieck (mit zwei gleich steilen Flanken). Das Testbit muß manchmal gesetzt und sogleich wieder gelöscht werden, um das Rauschen starten zu können. Ringmodulation und Synchronisation sind zwei Spezialeffekte, bei denen jeweils zwei Stimmen miteinander verknüpft werden. Die zugehörigen Tasten zu diesen Bits des Schwingungsform-Bytes sind in unserem Programm nebeneinander in der zweiten Reihe von unten auf der Tastatur. Die gleiche Taste schaltet abwechselnd das gleiche Bit ein und aus (aber erst, wenn das Programm mit dem Ändern des Bildes fertig ist, wird die Tastatur wieder abgefragt !)

Die Pulsbreite ist das Verhältnis von "Oben" und "Unten" in der Schwingungsform "Rechteck": der Klang ist an beiden Enden gleich, in der Mitte aber anders. Unser Programm verändert die Pulsbreite in wesentlich gröberen Schritten, als es beim C 64 möglich ist (es nutzt nur 4 von 12 Bits !). Zum Verändern werden die beiden Tasten ← und ↑ (links oben) benutzt. Auch für die Oktavenumschaltung gibt es zwei Tasten: den Punkt und den Schrägstrich /. Beachten Sie bitte, daß die oberste Oktave nur unvollständig zu Gehör gebracht werden kann, und daß sich die Oktaven gegenseitig überlappen.
Der Doppelpunkt schaltet die jeweilige Stimme auf das eingebaute Filter, dessen Eigenschaften ansonsten für alle Stimmen gemeinsam verändert werden können: Es kann als Hochpaß (HP,

hohe Töne durchgelassen), Bandpaß (BP, mittlere durchgelassen) und Tiefpaß geschaltet werden; mehrere gleichzeitig lassen entsprechend mehr durch (!). Nahe benachbart ist das Bit zum Stummschalten der 3. Stimme: mit der Funktionstaste f1 wird es an- oder ausgeschaltet (sinnvoll z.B. bei SYNC oder RING). Die Frequenz, auf die sich die Durchlaßeigenschaften (HP,BP und TP und die Resonanzstärke) beziehen, ist hier FF genannt; sie wird mit dem Semikolon kleiner oder mit dem Gleichheitszeichen größer gemacht. Die Resonanzstärke RS wird mit den Tasten K und L gesteuert. Die beiden Cursortasten schließlich bestimmen über die gesamte Lautstärke LS (leider für alle drei Stimmen gemeinsam; einen Sinn hat das eigentlich nur, wenn man ein Fernsehgerät außerhalb der Reichweite der Arme ohne Fernbedienung hat).

Sie sehen: Die Tastatur ist fast voll ausgenutzt (sogar fast ganz dreifach belegt !), und der Bildschirm ist ziemlich voll, jedenfalls mit Plätzen für bestimmte Informationen. Das erste Bild zeigt den Zustand beim Starten des Programms (durchaus zum Spielen geeignet), das zweite Bild zeigt etwas mehr, ist aber in dieser Zusammenstellung nicht zu empfehlen.

Ganz rechts unten in der Ecke können Sie nachsehen, wie die oberen beiden Tastenreihen als Klaviertasten zugeordnet sind: Q ist C, 2 ist Cis usw. (übrigens ein Beispiel für die Flexibilität der Texteinblendung in die Hochauflösung bei SIMON's BASIC mit Verwendung der (hochauflösenden Hard-)COPY).

DIE ORGANISATION DES SID (Sound Interface Device)

Der Speicherbereich von 54272 bis 54296 (also ganze 25 Bytes) steuert die dreistimmige Musik. Die Tabelle gibt einen Überblick über die Aufteilung der einzelnen Funktionen auf diese Bytes: Teilweise ist jedes Bit mit einem anderen Aspekt befaßt, teilweise erstreckt sich eine genaue Zahl über zwei Bytes. Am linken Rand sind die echten Adressen für die drei Stimmen angegeben (oben) bzw. für alle Stimmen gemeinsam (unten), am rechten Rand sind die gleichen Adressen relativ

Adresse für Stimme 1	2	3	Bit Nr. 7 / Bit Wert 128	6 / 64	5 / 32	4 / 16	3 / 8	2 / 4	1 / 2	Ø / 1	
54272	54279	54286	128	64	32	16	8	4	2	1	Ø 7 14
			F r e q u e n z								
54273	54280	54287	32768	16384	8192	4096	2048	1024	512	256	1 8 15
54274	54281	54288	128	64	32	16	8	4	2	1	2 9 16
							T a s t v e r h ä l t n i s				
54275	54282	54289					2048	1024	512	256	3 10 17
54276	54283	54290					Test-Bit	Ring-modul.	Syn-chron.	Ein/Aus	4 11 18
54277	54284	54291	8	4	2	1	8	4	2	1	5 12 19
			A t t a c k				D e c a y				
54278	54285	54292	8	4	2	1	8	4	2	1	6 13 20
			S u s t a i n				R e l e a s e				
		54293						4	2	1	21
							F i l t e r - f r e q u e n z				
		54294	1024	512	256	128	64	32	16	8	22
		54295	8	4	2	1	Filter wirkt auf: ext.Eing	St.3	St.2	St.1	23
			Resonanzstärke								
		54296	3 aus	Hochpaß	Bandpaß	Tiefpaß	8	4	2	1	24
							Lautstärke allgemein				

zur Basisadresse 54272 angeschrieben. Die acht Spalten der Tafel entsprechen den acht Bits in einem jeden Byte, von rechts nach links von Ø bis 7 numeriert. Im Inneren der Tabelle finden Sie nun die Bedeutungen der einzelnen Bits: was sie beeinflussen, und welchen Zahlenwert sie dabei darstellen. Die Zahlen sind also bei auf Eins gesetzten Bits innerhalb des gleichen eingerahmten Kästchens zu addieren.

Beachten Sie bitte, daß natürlich die POKE-Anweisung nur auf das ganze Byte (also z.B. 16*ATTACK+DECAY) angewendet werden kann, und daß außerdem die Speicherplätze des Sound Interface Device nur beschrieben, aber nicht sinnvoll mit PEEK gelesen werden können: Es empfiehlt sich daher evtl., ihre Inhalte in Form einer indizierten Variablen zu speichern. Das ist insbesondere für die Register mit den verschiedenen Schwingungsformen wichtig: Das Bit zum Ein- und Ausschalten muß geändert werden, ohne daß die Schwingungsform selbst gelöscht wird; der Computer weiß sonst z.B. nicht, mit welcher Schwingungsform er RELEASE ausführen soll, wenn Sie beim Abschalten eines Tones das ganze Byte auf Null setzen.

Eine Version ohne SIMON

Zum Schhluß des Musik-Abschnitts nun noch eine SYNTHESIZER-Version ohne SIMON's BASIC. Die zugehörigen Bildschirmgrafiken erscheinen hier mit Zeilenzwischenräumen, die auf dem Bildschirm fehlen. Das obere Bild gilt für den Start des Programms, das andere ist ein musikalisch unsinniges Beispiel mit vielen Zeichen.

```
         STIMMEN  -  SH   C=   CT   +

          ATTACK (Z)|    |▮   |▮   |(X)
           DECAY (C)|    |▮   |▮   |(V)
         SUSTAIN (B)▬▬   |▬▬▬ |▬▬▬ |(N)
         RELEASE (M)|    |▮   |▬   |(,)

        RAUSCHEN (A)                (A)
        RECHTECK (S)
       SAEGEZAHN (D)                (D)
         DREIECK (F)                (F)
         TESTBIT (G)                (G)
      RINGMODUL. (H)                (H)
     SYNCHRONIS. (J)                (J)
       TASTVERH. (←)|    ||   ||   |(1)
          OKTAVE (.)                (/)
          FILTER (:)                (:)

     3 (F1)        FILTERFREQU.(;)▬▬  |(=)
     HP(F3)        RESONANZ    (K)▮   |(L)
     BP(F5)
     LP(F7)

       LAUTSTAERKE (DOWN)▬▬▬ (RIGHT)
```

```
         STIMMEN  -  SH   C=   CT   +

          ATTACK (Z)▬▬   |▮   |▮   |(X)
           DECAY (C)▮    |▬▬  |▬▬  |(V)
         SUSTAIN (B)▬▬   |▬▬▬ |▬▬▬▬|(N)
         RELEASE (M)▮    |▮   |▬   |(,)

        RAUSCHEN (A)                (A)
        RECHTECK (S)                (S)
       SAEGEZAHN (D)                (D)
         DREIECK (F)                (F)
         TESTBIT (G)TEST            (G)
      RINGMODUL. (H)     RING       (H)
     SYNCHRONIS. (J)           SYNC (J)
       TASTVERH. (←)|    ||   ||   |(1)
          OKTAVE (.)                (/)
          FILTER (:)                (:)

     3 (F1)        FILTERFREQU.(;)▬▬  |(=)
     HP(F3)        RESONANZ    (K)▮   |(L)
     BP(F5)
     LP(F7)

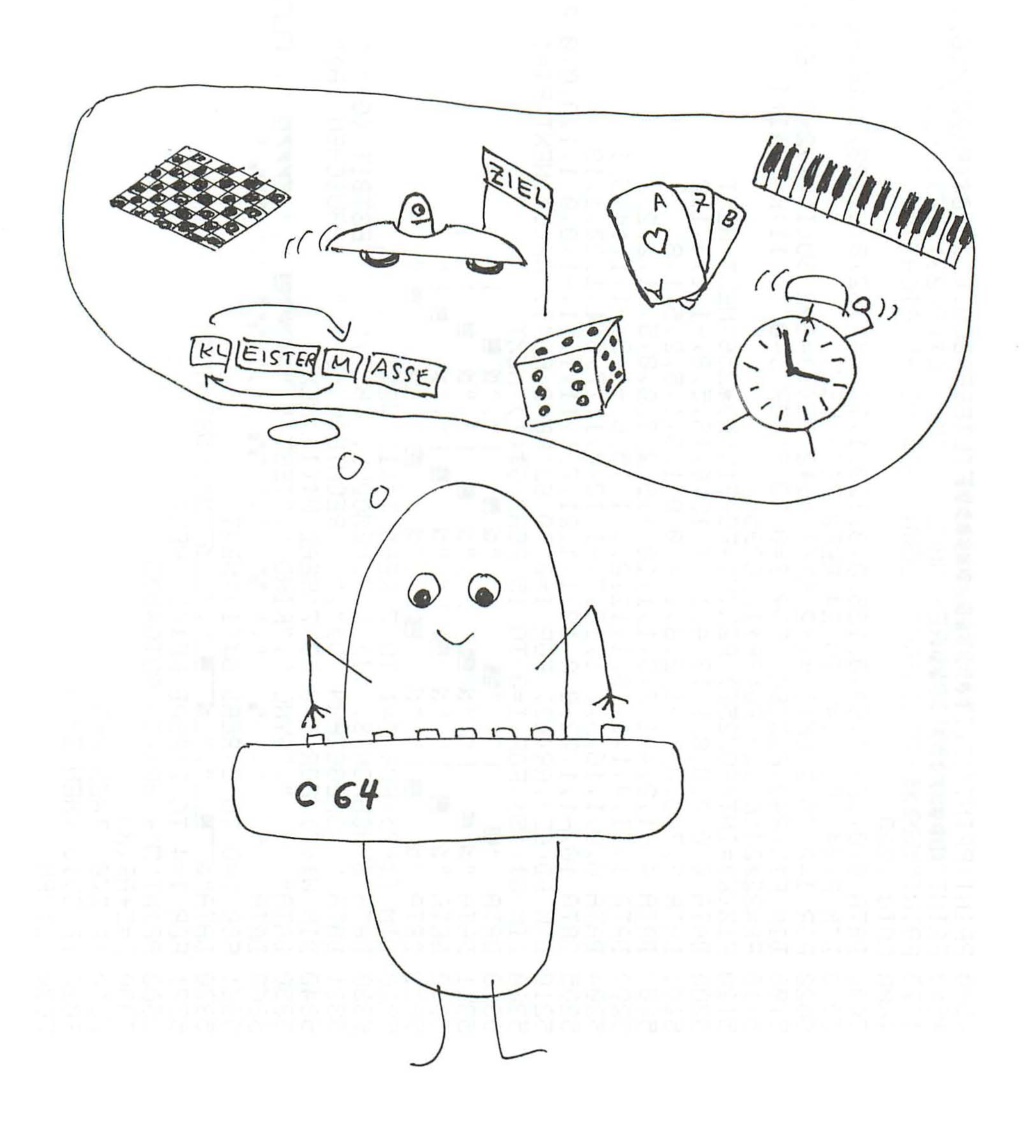
       LAUTSTAERKE (DOWN)▬▬  |(RIGHT)
```

```
1000 REM SYNTHESIZER
1001 :
1010 SI= 54272:PRINT"BITTE WARTEN !":GOTO 5000
1100 PRINT"▒":FOR ZZ=1 TO ZE:PRINT:NEXT:RETURN
1150 POKE SI+RR,R(RR):RETURN
1200 X=PEEK(203):YY=PEEK(653)
1210 IF X=64 THEN 1300
1220 IF YY=0 THEN 1200
1230 IF AD(X)=1 THEN 1500
1240 FOR I=0 TO 2:IF (YY AND 2↑I) THEN Y=I
1250 NEXT
1260 ON AD(X)GOTO1500,2200,2300,2400,2500,2600,2700,2800,2900,3000,3100,3200,330
0
1300 FOR Y=0 TO 2:POKE SI+4+Y*7,R(4+Y*7):NEXT:GOTO 1200
1500 FOR Y=0 TO 2
1510 IF (YY AND 2↑Y) =0 THEN 1560
1520 POKE SI  +7*Y,F2%(12*OK(Y)+AR(X))
1530 POKE SI+1+7*Y,F1%(12*OK(Y)+AR(X))
1540 POKE SI+4+7*Y,R(4+7*Y) OR 1
1560 NEXT:GOTO 1200
2200 RR=4+7*Y:IF (R(RR) AND 2↑AR(X)) THEN R(RR)=R(RR)-2↑AR(X):GOSUB1150:GOTO 223
0
2210 RR=4+7*Y:IF (R(RR) AND 2↑AR(X))=0 THEN R(RR)=R(RR)+2↑AR(X):GOSUB1150:GOTO 2
240
2230 ZE=13-AR(X):GOSUB 1100:PRINTTAB(15+5*Y);W$(0):GOSUB 1150:GOTO 1200
2240 ZE=13-AR(X):GOSUB 1100:PRINTTAB(15+5*Y);W$(AR(X)):GOTO 1200
2300 RR=5+7*Y:Z=AK(Y):GOSUB7000:AK(Y)=Z:R(RR)=16*AK(Y)+DE(Y):G=AK(Y):ZE=1:GOTO 2
650
2400 RR=5+7*Y:Z=DE(Y):GOSUB 7000:DE(Y)=Z:R(RR)=16*AK(Y)+DE(Y):G=DE(Y):ZE=2:GOTO
2650
2500 RR=6+7*Y:Z=SU(Y):GOSUB 7000:SU(Y)=Z:R(RR)=16*SU(Y)+RE(Y):G=SU(Y):ZE=3:GOTO
2650
2600 RR=6+7*Y:Z=RE(Y):GOSUB 7000:RE(Y)=Z:R(RR)=16*SU(Y)+RE(Y):G=RE(Y):ZE=4
2650 GOSUB 1150:GOSUB 1100:PRINTTAB(15+5*Y)G$(G):GOTO 1200
2700 OK(Y)=OK(Y)+AR(X)
2710 IF OK(Y)<0 THEN OK(Y)=0
2720 IF OK(Y)>5 THEN OK(Y)=5
2730 ZE=14:GOSUB 1100:PRINTTAB(15+Y*5)O$(OK(Y));:GOTO 1200
```

```
2800 Z=VO:GOSUB 7000:VO=Z:R(24)=(R(24) AND 240)+VO:RR=24:GOSUB 1150
2810 ZE=22:GOSUB 1100:PRINTTAB(20)G$(VO);:GOTO 1200
2900 Z=RS:GOSUB 7000:RS=Z:R(23)=16*RS+FI:ZE=19:RR=23:GOSUB 1150
2910 GOSUB 1100:PRINTTAB(29)G$(R(23)/16):GOTO 1200
3000 Z=RF:GOSUB 7000:RF=Z:R(22)=16*RF:ZE=18:RR=22:GOSUB 1150
3010 GOSUB 1100:PRINTTAB(29)G$(R(22)/16):GOTO 1200
3100 RR=3+7*Y:Z=PU(Y):GOSUB 7000:PU(Y)=Z:R(RR)=PU(Y):ZE=13:GOSUB 1150
3110 GOSUB 1100:PRINTTAB(15+5*Y)G$(PU(Y)):GOTO 1200
3200 RR=23:IF (FI AND 2↑Y)=2↑Y THEN FI=FI-2↑Y:GOTO 3220
3210 IF (FI AND 2↑Y)=0   THEN FI=FI+2↑Y:GOTO 3230
3220 R(23)=FI+16*RS:ZE=15:GOSUB 1100:PRINTTAB(15+5*Y)"     ":GOSUB 1150:GOTO1200
3230 R(23)=FI+16*RS:ZE=15:GOSUB 1100:PRINTTAB(15+5*Y)"▛FILT■":GOSUB 1150:GOTO1200
3300 RR=24:IF (R(24) AND 2↑AR(X)) THEN R(24)=R(24)-2↑AR(X):GOTO 3350
3310 IF (R(24) AND 2↑AR(X))=0 THEN R(24)=R(24)+2↑AR(X):GOTO 3360
3350 ZE=24-AR(X):GOSUB 1100:PRINTTAB(6)"      ":GOSUB 1150:GOTO 1200
3360 ZE=24-AR(X):GOSUB 1100:PRINTTAB(6)GF$(AR(X)):GOSUB 1150:GOTO 1200
4000 FOR I=0 TO 24:POKE SI+I,R(I):NEXT
4100 PRINT"▛  STIMMEN  -  SH   C=   CT   +▛"
4110 PRINT"    ATTACK (Z)";:FOR Y=0 TO 2:PRINT G$(R(5+7*Y)/16);:NEXT:PRINT"(X)"
4120 PRINT"     DECAY (C)";:FOR Y=0 TO 2:PRINT G$(R(5+7*Y)AND15);:NEXT:PRINT"(V
)"
4130 PRINT"   SUSTAIN (B)";:FOR Y=0 TO 2:PRINT G$(R(6+7*Y)/16);:NEXT:PRINT"(N)"
4140 PRINT"   RELEASE (M)";:FOR Y=0 TO 2:PRINT G$(R(6+7*Y)AND15);:NEXT:PRINT"(,
)▛"
4200 FOR I=7 TO 1 STEP-1
4210 PRINTT$(I);:FOR Y=0 TO 2:G$=W$(0)
4220 IF (R(4+7*Y) AND 2↑I) THEN G$=W$(I)
4230 PRINT G$;:NEXT:PRINT RIGHT$(T$(I),3):NEXT
4300 PRINT"  TASTVERH. (←)";:FOR Y=0 TO 2:PRINTG$(PU(Y));:NEXT:PRINT"(1)"
4305 PRINT"    OKTAVE (.)";:FOR Y=0 TO 2:PRINT O$(OK(Y));:NEXT:PRINT"(/)"
4310 PRINT"    FILTER (:)";:FOR Y=0 TO 2:G$="      "
4320 IF (FI AND 2↑Y) THEN G$="▛FILT■ "
4330 PRINTG$;:NEXT:PRINT"(;)"
4400 PRINT"▛3 (F1)"
4410 PRINT"HP(F3)      ";:IF (R(24) AND 64) THEN PRINT"■■■■L_▛‾‾";
4420 PRINT:PRINT"BP(F5)      ";:IF (R(24) AND 32) THEN PRINT"■■■■L_▛‾‾▀▄_";
4430 PRINT:PRINT"LP(F7)      ";:IF (R(24) AND 16) THEN PRINT"■■■■■▛‾‾▀▄_";
```

```
4500 PRINT:PRINT"FILTERFREQU.(;)";G$(R(22)/16);"(=)"
4510 PRINT"RESONANZ     (K)";G$(R(23)/16);"(L)"
4520 PRINT"LAUTSTAERKE (DOWN)"+G$(VO)+"(RIGHT)";
4600 GOTO 1200
5000 DATA 0,0,128,1,64,0,128,0,0,128,1,32,34,162,0,0,128,1,16,68,200,0,128,41,79
5010 DIM R(24):FOR I=0 TO 24:READ R(I):NEXT
5020 FOR I=0 TO 2:OK(I)=I+2:AK(I)=I*2:DE(I)=I*2:SU(I)=I*2+8:RE(I)=I*2:NEXT:VO=15
5100 DIM F1%(84),F2%(84):FOR I=0 TO 5:FOR J=0 TO 11:K=12*I+J
5110 FR=32*2↑(K/12):FQ=FR*17.0295
5120 F1%(K)=INT(FQ/256):F2%(K)=FQ-F1%(K)*256:NEXT:NEXT
5200 DATA 0,0,0,0,8,1,13,4,13,7,13,6,13,5,8,-1,1,3,1,2
5201 DATA 2,7,0,0,3,-1,2,6,1,4,0,0,1,6,1,5,2,5,1,8
5202 DATA 4,-1,2,4,1,7,3,1,1,10,1,9,2,3,0,0,5,-1,2,2
5203 DATA 1,11,4,1,1,13,1,12,2,1,1,15,6,-1,9,-1,1,14,5,1
5204 DATA 0,0,1,16,9,1,1,18,7,-1,12,0,1,17,6,1,1,20,1,19
5205 DATA 10,-1,1,22,0,0,10,1,1,21,7,1,11,1,11,-1,0,0,1,1,0,0,0,0,1,0
5210 DIM AD(62),AR(62):FOR I=0 TO 62:READ AD(I),AR(I):NEXT:FI=1
5300 DIM G$(15):FOR I=0 TO 15:READ G$(I):NEXT
5310 DATA "     |","     |","     |","     |"
5311 DATA "     |","     |","     |","     |"
5312 DATA "     |","     |","     |","     |"
5313 DATA "     |","     |","     ","     "
5320 DIM T$(7):FOR I=1 TO 7:READ T$(I):NEXT
5330 DATA SYNCHRONIS. (J)," RINGMODUL. (H)","   TESTBIT (G)","    DREIECK (F)"
5331 DATA "  SAEGEZAHN (D)","  RECHTECK (S)","  RAUSCHEN (A)"
5340 DIM W$(7):FOR I=0 TO 7:READ W$(I):NEXT
5350 DATA"      ","SYNC ","RING ","TEST ","     ","     ","     ","     "
5360 DATA "     ","     ","     ","     ","     ","     "
5361 FOR I=0 TO 5:READ O$(I):NEXT
5370 DATA"     ","     ","     ","STUMM"
5371 FOR I=4 TO 7:READ GF$(I):NEXT
5500 PRINT""+CHR$(8):GOTO4000
7000 Z=Z+AR(X)
7010 IF Z<0  THEN Z=0
7020 IF Z>15 THEN Z=15
7030 RETURN
```

ANREGUNGEN FÜR SPIELE UND PROGRAMME



DIE ZEIT IM BILD

Darstellungen der Zeit

Das Schöne an einem Bildschirmcomputer ist im Vergleich zu älteren Rechnern, die nur Ergebnisse auf Papier ausdrucken konnten, daß Vorgänge in ihrem echten zeitlichen Ablauf, wie in einem Film, gezeigt werden können (natürlich meist in Dehnung oder in Raffung). Das sollte man auch dann ausnutzen, wenn nur eine Größe als zeitlich veränderliche Größe gezeigt werden soll, etwa als waagerechter oder senkrechter Balken entsprechender Länge. Damit man den zeitlichen Verlauf auch rückblickend vor sich hat, läßt man dann die Zeit in der zweiten Richtung ablaufen. Besonders einfach geht das bei der mit PRINT erzeugten Grafik mit waagerechten Balken: Wird der untere Rand erreicht, schiebt sich das Bild von selbst nach oben weiter, und mit Tastaturgrafik kann man mit SIMON's BASIC den Bildschirminhalt nach allen Seiten durchschieben (mit LEFT RIGHT UP und/oder DOWN). Arbeitet man aber in der HIRES-Feingrafik, so geht das leider nicht. Wenn die Zeit von links nach rechts aufgetragen wird, kann man beim Erreichen des rechten Randes das ganze Bild auf einen Schlag löschen, etwa so:

```
9ØØ IF X  319 THEN X=Ø:HIRES 1,Ø
```



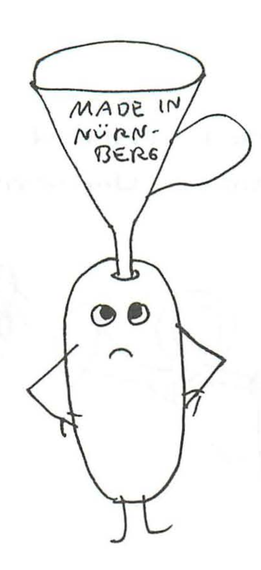
Dann ist aber jedesmal eine Zeitlang nicht mehr zu sehen, was kurz vorher gewesen ist. Sie können auch eine LINE-Anweisung verwenden, die ein Stück rechts von der jeweiligen Zeichenposition einen senkrechten Streifen löscht. Das erinnert an einen Lehrer, der auf einer vollen Tafel von links nach rechts etwas anschreibt und dabei mit der linken Hand immer rechts von seiner rechten mit dem Lappen etwas leerwischt. Im Programm kann das etwa so aussehen:

```
100 HIRES 1,6:XX=20
110 Y=Y*.99+.2*RND(1)-.1
120 LINE XX,0,XX,199,0
130 LINE X,100+Y*100,X,199,2
140 PLOT X,100,2
150 X = X+1:IF  X=320 THEN  X=0
160 XX=XX+1:IF XX=320 THEN XX=0
170 GOTO 110
```

QUIZ UND DATENSPEICHER

Abfragen als Spiel und als Service

Wenn Sie viele Daten über ein bestimmtes Thema in ein Programm schreiben (oder in einer sequentiellen Datei unterbringen, was etwas umständlicher ist), können Sie sie entweder auf Stichwörter hin abfragen; Sie können aber auch ein Quiz daraus machen, bei dem der Computer (evtl. nach vorgegebenen Kriterien wie Teilthemen, Schwierigkeit usw.) Stichwörter bzw. Fragen durch Zufall auswählt und ausgibt. Der Spieler muß dann eine richtige Antwort eingeben, bekommt dann Plus- oder Minuspunkte, sein Punktekonto, eine Korrektur der falschen Antwort oder ein Lob für eine richtige. Das ist zwar vielleicht nicht die lustigste Art, mit einem Computer umzugehen, aber andererseits immer noch eine relativ unterhaltsame Art, Vokabeln zu lernen oder Grammatik zu üben (vor allem wenn man das Programm selbst schreibt, anhand eines Lehrbuches).
Die Daten können natürlich auch grafisch veranschaulicht werden. Enthält ein Programm verschiedene Zahlenangaben über die Staaten (auch Zugehörigkeit zu Erdteilen oder Organisationen), kann man sich z.B. die Pro-Kopf-Bruttosozialprodukte der 10



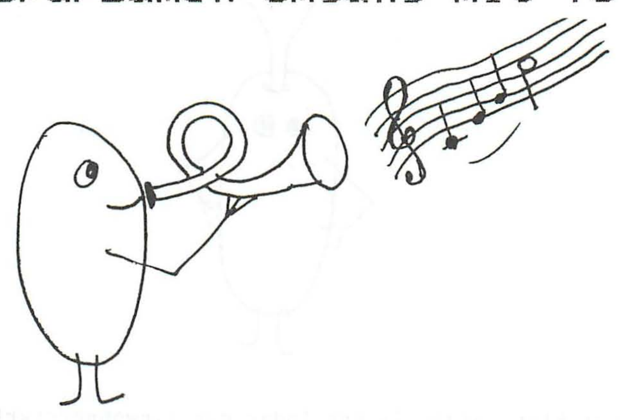
flächengrößten Länder Asiens (oder der einwohnerstärksten Länder der NATO usw.) als Säulendiagramm (Histogramm) zeichnen lassen; Sie können sogar (wenn in den Daten jährliche Wachstumsraten enthalten sind), dasselbe für spätere Jahrzehnte machen, natürlich nur im Rahmen der Gültigkeit solcher Wachstumsraten.

WOZU MULTICOLOR ?
Auflösung gegen Mehrfarbigkeit

Beim VC 2Ø ist oft die Entscheidung zwischen voller Auflösung (16Ø x 16Ø Punkten) mit sehr bescheidener Farbdifferenzierung (nur 2ØØ einzeln färbbaren REGIONs) und der MULTIcolor (8Ø x 16Ø einzeln zu färbenden Punkten mit 4 Farben zur Auswahl) sehr schwer. Beim C 64 sind wir in beiden Optionen deutlich besser daran: Bei Hochauflösung (32Ø x 2ØØ Punkten) gibt es immerhin 1ØØØ Farbregionen (mit POKE 48*1Ø24+...), und andererseits bietet ja hier die MULTIcolor noch eine vernünftige Auflösung: 16Ø x 2ØØ. In manchen Fällen ist es trotzdem nicht gleichgültig, welche der beiden Möglichkeiten man verwendet: MULTIcolor ist insbesondere dann besser, wenn es auf Annäherungen oder Überkreuzungen verschiedener Kurven ankommt, bei denen man möglichst an jedem einzelnen Punkt erkennen möchte, zu welcher Kurve er gehört. Überlagerungen von Wellen, Modulationen, Näherung einer Funktionskurve durch eine andere (oder durch Summen von anderen, z.B. Fourier-Reihen) sind da wichtige Beispiele.

PIEPT ES BEI IHNEN DAUERND ?

Vom sparsamen Umgang mit Tönen



Obwohl der C 64 einen veritablen Synthesizer eingebaut hat, sind reine Musikprogramme auf ihm eher spielerische Demonstrationen seiner Möglichkeiten als wirklicher Ersatz für eine Orgel oder einen Profi-Synthesizer. Seine Stärke ist vielmehr die Kombination der Musik mit dem Bild: So kann man sie spielen (lassen) und melodische oder harmonische Strukturen sichtbar darstellen, während sie erklingen, seien sie nun vorprogrammiert, zufallserzeugt oder "live" auf den Tasten gespielt - oder auch aus diesen Methoden gemischt. So kann dann eine Zufallsmusik entstehen mit vorprogrammierten oder zwischendurch eingegebenen Wahrscheinlichkeitswerten für Töne oder Intervallschritte usw.

Töne können auch Spiel- oder Vorführprogramme akustisch kommentieren oder in ihnen Informationen ausgeben. Ähnlich wie bei einer modernen Registrierkasse können Eingaben zur Bestätigung mit einem Piepton quittiert werden, bei Auto-Rennspielen kann z.B. die Geschwindigkeit durch die Tonhöhe angezeigt werden, beim Erreichen des Zieles wird eine Fanfare (aufsteigender Dreiklang z.B.) geblasen, beim CRASH kommt der Rauschgenerator zum Einsatz (mit abgleitender Frequenz). Es wird aber (vor allem wenn der Spieler noch ungeschickt ist) bald sehr nervtötend, wenn bei jedem Mißerfolg Chopins Trauermarsch ertönt: weniger ist oft besser.

WORTSPIELE

Textverarbeitungsprogramme sind besonders dann interessant, wenn sie eher spielerischen Charakter haben; dazu einige Anregungen:

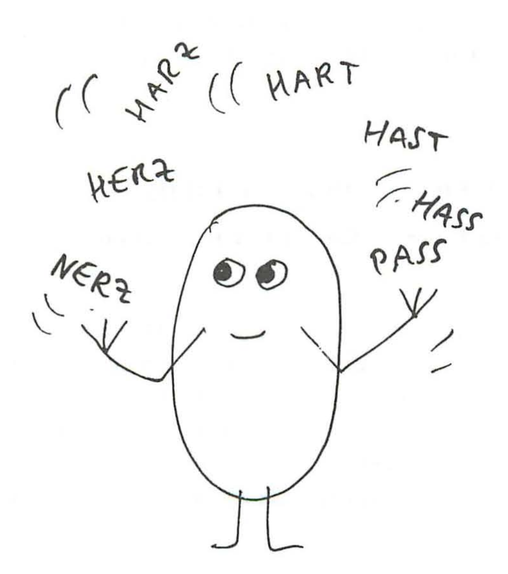
Eingetippte Buchstaben ertönen sofort oder auf Abruf als akustische Morsezeichen,

eingegebene Wörter werden durchgeschüttelt (Anagramme), oder es werden in vorgeschriebener Dosis Druckfehler eingestreut (Test auf Lesbarkeit),

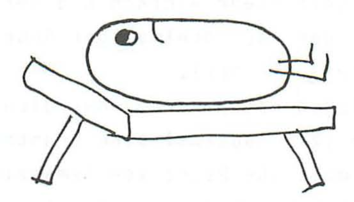
ein Text wird in Geheimschrift verschlüsselt oder gar entschlüsselt (sehr einfach und verblüffend ist ein Code, der nach dreimaliger Anwendung wieder den Klartext liefert),

mit dem Zufallsgenerator werden Parodien auf exotische Tiernamen (z.B. australische Drahthaar-Natter) oder auf modische Redetexte (emanzipatorisches Katastrophen-Bewußtsein) erzeugt,

im Dialog werden Symptome für eine (scherzhafte) medizinische Diagnose oder eine Wetterprognose eingeholt.



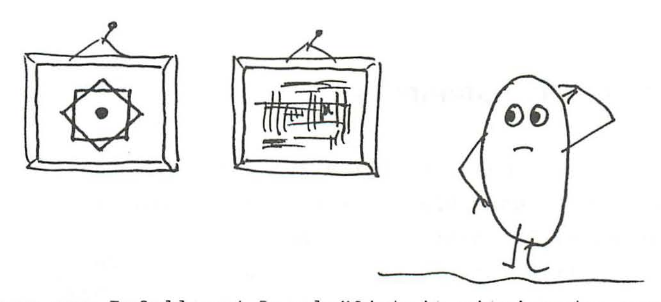
Man kann sich sicher über die literarische Qualität von computererzeugter Lyrik streiten (im Vergleich zu anderer moderner Lyrik natürlich), aber immerhin ist bemerkenswert, daß der C 64 ohne weiteres witzige Zusammenstellungen von Wörtern bringt, ohne selbst ihre Witzigkeit feststellen zu können; er ähnelt darin dem Kindermund. Auch sollte hier das Programm ELIZA (benannt nach der zur Sprache erweckten Eliza Doolittle aus Shaws Pygmalion) von Joseph Weizenbaum erwähnt werden: es parodiert einen dialogführenden Psychiater, der stets sehr "verständnisvoll" auf die Äußerungen des Patienten eingeht. Im Programm läuft das mit INPUT und PRINT (eine BASIC-Version ist in dem im Literaturanhang erwähnten Buch "More BASIC Com-



puter Games" enthalten). Weizenbaum war bestürzt über die Bereitschaft der Menschen, dem Computer wider besseres Wissen menschliche Qualitäten zuzuschreiben und ihm Vertrauen entgegenzubringen; sogar einige Psychiater wollten das Programm ernsthaft für die klinische Anwendung erweitern und einsetzen.

IST DAS ETWA KUNST ?
Computer-Grafik zum Anschauen

Da gibt es zum Beispiel die Viertelkreise in der Tastatur-Grafik. Schreibt man mit ihnen den Bildschirm ganz zufällig voll, so ist das Ergebnis ziemlich chaotisch. Schreibt man sie in einer regelmäßigen Reihenfolge, so entsteht ein mehr oder weniger interessantes Muster, dessen ästhetischen Wert man in der Nähe von Tapeten einordnen kann. Spannend wird es,



wenn man Zufall und Regelmäßigkeit miteinander verknüpft, z.B. so, daß keine Linie ins Leere mündet, sondern nur geschlossene Kurven entstehen (abgesehen vom Rand natürlich). Das ist wohl das allgemeinste Kriterium für Kunst: Die Information darf nicht zu groß (Chaos) und nicht zu klein (leeres Bild) sein. Bei einem Rätsel wird das klar: damit es interessant ist, darf es nicht zu leicht sein ("Wieviele Tage hat die Woche ?"), aber es darf auch nicht unlösbar schwer sein ("Welche Nummer steht im Telefonbuch unserer Stadt auf Seite 1ØØ als erste ?"); damit es reizvoll ist, ein Rätsel zu lösen, muß man eine gewisse Chance haben, aber sie darf nicht Eins sein. Bei einer Melodie ist es ähnlich: Wenn man in jedem Moment genau weiß, wie sie weitergeht, ist sie langweilig und abgedroschen; kann man überhaupt nichts vorhersagen, gibt man das Raten bald auf. Am meisten Freude hat man, wenn man ungefähr weiß, wie es weitergeht, aber doch in Feinheiten immer etwas überrascht wird. Das Zusammenspiel von Überraschung und Bestätigung scheint eine große Rolle zu spielen. Für reizvolle Computergrafik (wie auch für Computermelodien und Computerlyrik) ergibt sich daraus die einfache Grundregel: Der Zufallsgenerator muß vorkommen, aber er muß durch regelmäßige Strukturen oder Vorschriften ergänzt werden. Anders gesagt: Die Freiheit des Zufalls muß in nicht zu enge Bahnen gelenkt werden.

Wie wenig Information in einfachen Strichzeichnungen steckt, erkennt man sehr schön anhand von Programmen, die etwa in der Art von Fahndungs-Phantombildern Gesichter oder Wolpertinger (gemischte Fabeltiere) herstellen: Eine Zahl (von Ø bis 7) gibt zum Beispiel die Länge des Halses an, eine andere die Augengröße usw.

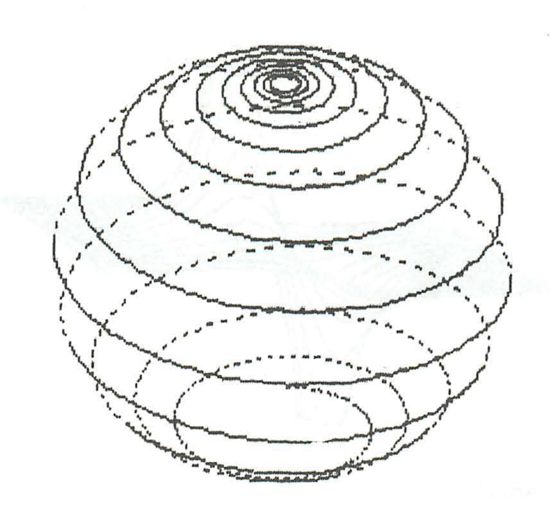
SPIELE MIT GRAFIK

Wie viele Bilder kann man eigentlich auf dem C 64 machen ? Es gibt 64000 Punkte und 1000 Farbregionen mit jeweils 16 Farben zur Auswahl. Das gibt für die Besetzung der Rasterpunkte 2^{64000} Möglichkeiten und für die Farben 16^{1000}, insgesamt also das Produkt aus diesen beiden Zahlen: 2^{68000} oder $10^{20470,0397}$ Möglichkeiten. Diese Zahl ist im Dezimalsystem eine 1 mit über 20000 Nullen. Zum Vergleich: Die Zahl der Atome im Weltall hat rund 80 Nullen, und wesentlich größere Zahlen kommen in der Physik nicht vor, auch nicht in der Astronomie, wenn man nicht gerade theoretische Möglichkeiten abzählt. Sie haben also sicher Verständnis dafür, daß hier nicht alle möglichen Bilder abgedruckt werden, sondern nur einige Beispiele (eine etwas umfangreichere Auswahl meiner Bildprogramme erscheint bei Hagemann). Ganz allgemein können Sie die Programme verändern, indem Sie Zahlen (nicht zuviele auf einmal) ändern oder probeweise einzelne Faktoren wegstreichen. Auf diese Weise dringen Sie auch leicht in die analytisch-geometrischen Zusammenhänge ein, die oft mit verblüffend einfachen Formeln raffinierte Bilder erzeugen können. Ein spezieller Trick ist die Benutzung des Zufallsgenerators zur Erzeugung von Zwischenstufen zwischen Schwarz und Weiß.

Das Ausführungstempo der Grafikprogramme ist sehr unterschiedlich: Einige hundert Linien oder Blocks dauern nur wenige Sekunden; ein Bild, bei dem hingegen 64000 oder noch mehr Entfernungen berechnet werden müssen, kann Stunden bis Tage dauern (abends eingeben, morgens anschauen und fotografieren oder ausdrucken lassen !).

Die folgenden Beispiele sind mit HIRES in SIMON's BASIC geschrieben und mit COPY ausgedruckt.

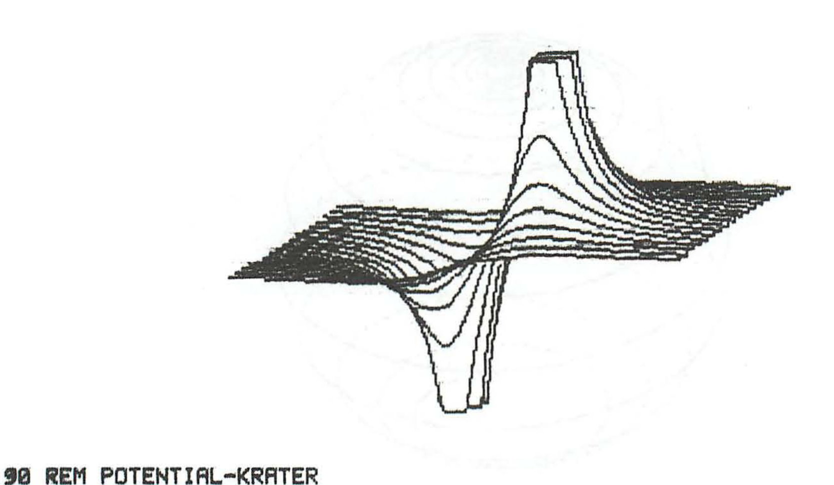
Wenn man auf einer Kugel beschließt, immer in die gleiche Richtung zu gehen, so kann das entweder bedeuten, daß man so geradeaus wie möglich geht, also auf einem Großkreis. Wenn man sich aber nach einem Kompaß richtet und immer den gleichen Winkel gegen die Nordrichtung beibehält, nähert man sich einem der beiden Pole (wenn man nicht gerade auf dem Äquator spaziert) auf einer Loxodrome. Im Schrägbild sieht das so aus (Wegteile auf der unsichtbaren Kugelhälfte gestrichelt !):



```
90 REM LOXODROME
91 :
100 HIRES 1,0:B=-1.2:W=.06:D=.6:XA=160:XB=160:YB=YA
101 YA=100-100*SIN(B)*COS(D)-100*COS(L)*COS(B)*SIN(D)
110 L=L+.1:B=B+W*.1*COS(B)
120 X=160+100*SIN(L)*COS(B)
130 Y=100-100*SIN(B)*COS(D)-100*COS(L)*COS(B)*SIN(D)
131 Z=SIN(B)*SIN(D)-COS(L)*COS(B)*COS(D)
140 IF Z>0 THEN LINE XA,YA,X,Y,1:XA=X:YA=Y
141 IF Z<=0 THEN LINE (2*X+XA)/3,(2*Y+YA)/3,X,Y,1:XA=X:YA=Y
145 GOTO 110

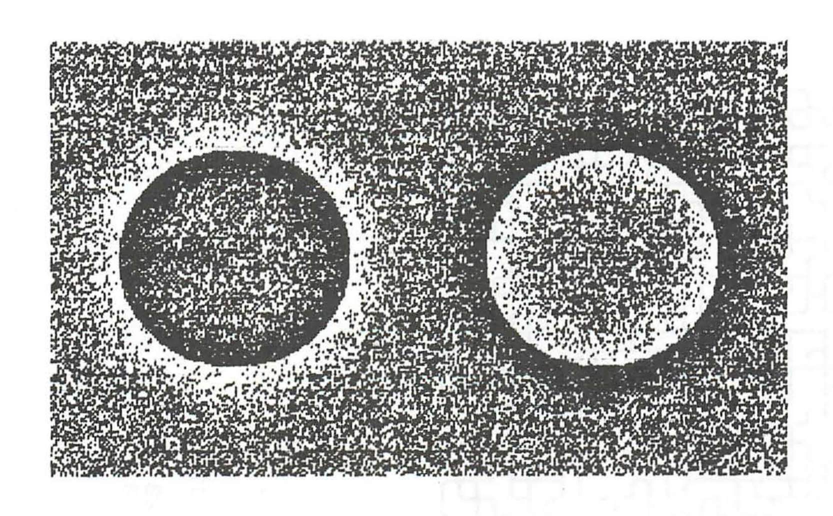
READY.
```

Eine Funktion von zwei Ortskoordinaten (oder sonstigen Variablen) kann man in drei Dimensionen als Gebirge darstellen. Aber wie bildet man das nun einigermaßen klar in zwei Dimensionen ab ? Eine Möglichkeit besteht darin, das Gebirge (in Gedanken) streifenweise anzumalen und diese Linien dann im Schrägbild zu zeigen. Wenn man es besonders raffiniert machen will, kann man auch dem Computer sagen, er möchte die gegenseitige Verdeckung berücksichtigen.
In unserem Beispiel handelt es sich um das elektrische Potential von zwei Punktladungen, wobei der Gipfel und der Krater abgeschnitten dargestellt sind:



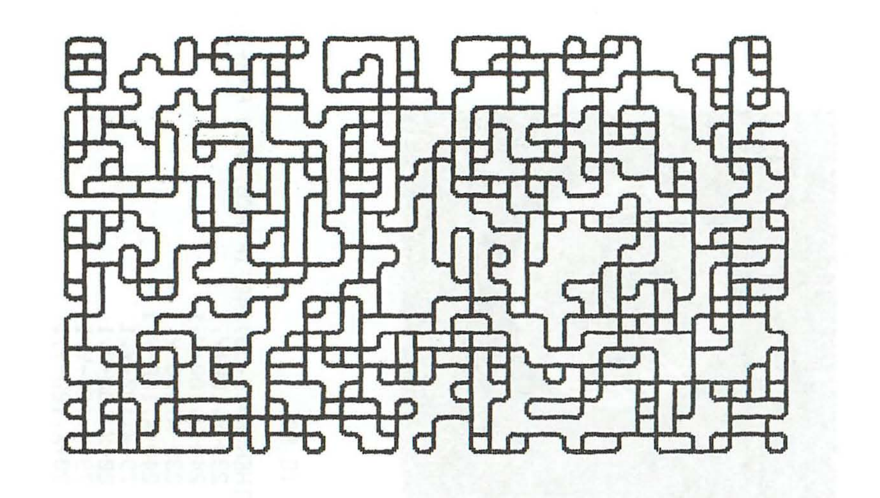
```
90 REM POTENTIAL-KRATER
91 :
100 HIRES 1,0:N=2:MA=1E4:DIM Z(320),ZZ(320)
101 FOR I=0 TO 320:ZZ(I)=199:NEXT
110 X(1)=80:Y(1)=100:Q(1)=1
111 X(2)=120:Y(2)=100:Q(2)=-1
120 FOR Y=50 TO 150 STEP 6
130 FOR X=1 TO 200
135 PO=0:FOR I=1 TO N
140 R(I)=SQR((X-X(I))↑2+(Y-Y(I))↑2):IF R(I)=0 THEN 200
310 PO=PO-Q(I)/R(I)*1E3:NEXT:IF PO>80 THEN PO=80
311 IF PO<-80 THEN PO=-80
312 IF PO>80 THEN PO=80
313 PO=PO+67+Y/3
320 IF PO>Z(X+Y/2) THEN GOSUB 400
321 IF PO<ZZ(X+Y/2) THEN GOSUB 410
350 NEXT:NEXT
360 GOTO 360
400 IF X=1 OR X=319 THEN Z(X+Y/2)=PO:RETURN
401 LINE X+Y/2,199-PO,X-1+Y/2,199-Z(X-1+Y/2),1:Z(X+Y/2)=PO:RETURN
410 IF X=1 OR X=319 THEN ZZ(X+Y/2)=PO:RETURN
411 LINE X+Y/2,199-PO,X-1+Y/2,199-ZZ(X-1+Y/2),1:ZZ(X+Y/2)=PO:RETURN
```

Glauben Sie, daß es mitten in den beiden Kreisen gleich hell ist ? Falls nicht, machen Sie sich eine Schablone, die nur die mittleren Teile der Kreise freiläßt !



```
100 HIRES 1,0
110 FOR X=0 TO 79:FOR Y=0 TO 99
120 A=.5:R=SQR(X↑2+Y↑2)
130 IF R>30 THEN IF R<70 THEN A=(R-50)/40:IF A<0 THEN A=A+1
150 IF A<RND(1) THEN PLOT 239-X,100+Y,1
151 IF A<RND(1) THEN PLOT 240+X,100+Y,1
152 IF A<RND(1) THEN PLOT 239-X,100-Y,1
153 IF A<RND(1) THEN PLOT 240+X,100-Y,1
154 IF A>RND(1) THEN PLOT 79-X,100+Y,1
155 IF A>RND(1) THEN PLOT 80+X,100+Y,1
156 IF A>RND(1) THEN PLOT 79-X,100-Y,1
157 IF A>RND(1) THEN PLOT 80+X,100-Y,1
160 NEXT:NEXT
170 COPY:COPY
300 GOTO 300
```

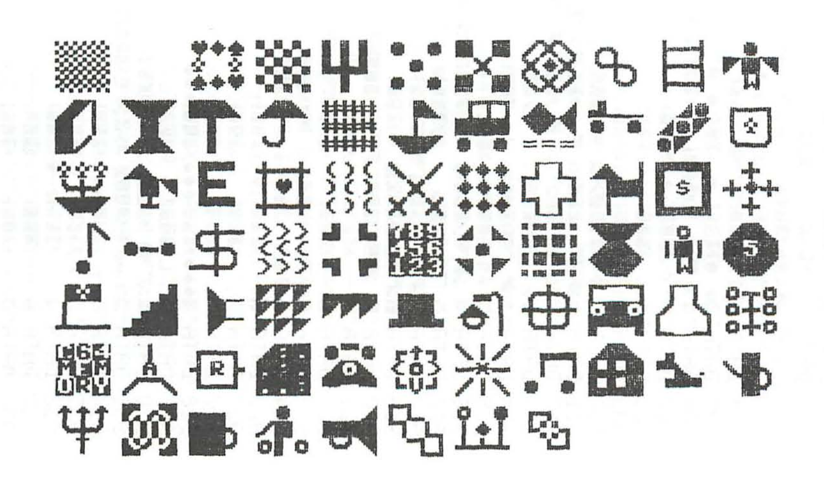
Das folgende Programm erzeugt ein Zufallsmuster aus Bögen, Kreuzen und T-Stücken mit der Randbedingung, daß (außer evtl. in der rechten unteren Ecke) keine Linien frei enden. Die Zeilen 1Ø1 und 1Ø2 stellen zwei Zeichensätze zur Auswahl: mit runden und mit eckigen Abbiegungen.



```
100 HIRES 15,0:DIM M%(40,25),A$(15)
101 DATA " ","*","*","┌","*","─","┐","┬","*","└","│","├","┘","┴","┤","┼"
102 REM    DATA " ","*","*","╭","*","─","╮","┬","*","╰","│","├","╯","┴","┤","┼"
103 FOR I=0 TO 15:READ A$(I):NEXT
110 FOR X=0 TO 39:FOR Y=0 TO 24
120 B=INT(16*RND(1)):IF A$(B)="*" THEN 120
130 IF (B AND 12)<>M%(X,Y) THEN 120
131 IF Y=24 THEN IF (B AND 2)=2 THEN 120
132 IF X=39 THEN IF (B AND 1)=1 THEN 120
140 TEXT X*8,Y*8,A$(B),1,1,8
150 IF (B AND 1)=1 THEN M%(X+1,Y)=(4 OR M%(X+1,Y))
160 IF (B AND 2)=2 THEN M%(X,Y+1)=(8 OR M%(X,Y+1))
190 NEXT:NEXT
200 GOTO 200
```

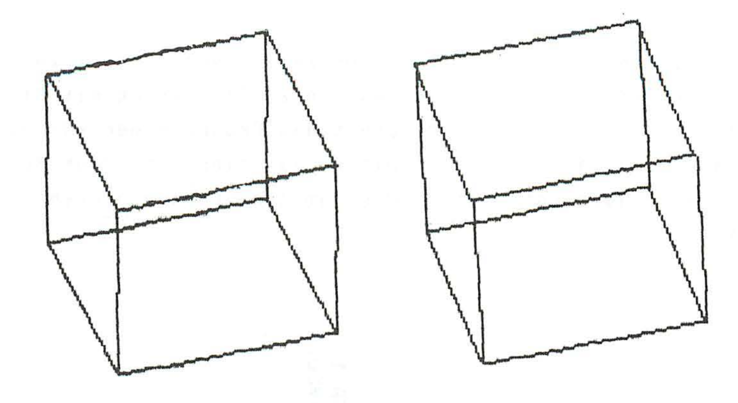
Spiele mit der Tastaturgrafik

Aus ganz alten Zeiten, als die Commodore-Computer noch keine Feingrafik hatten, stammt die großzügige Dreifachbelegung der Tasten mit Buchstaben und Grafikelementen. Man kann sie mit Farb- und REVERSE-Umschaltern und mit Cursorfunktionen kombinieren und auf diese Weise kleinere bunte Bilder als Strings definieren. Um Ihrer Phantasie auf die Sprünge zu helfen, ist hier der Bildersatz des MEMORY-Spieles * in Schwarz-Weiß und als Teil des LISTings wiedergegeben: Jede Zeile entspricht einem (meist mehrfarbigen) Bildchen aus 3 x 3 Buchstaben. Wollen Sie diese Zeichen in die HIRES-Grafik einbauen, so geht das mit TEXT oder mit CHAR, allerdings müssen Sie dann die Farben und die Cursorpositionen auf andere Weise realisieren; die REVERSE-Schaltung ist immerhin bei TEXT möglich, bei CHAR erfolgt sie durch Addition von 128.



*aus einer Spiele-Cassette des Autors im Hagemann-Verlag

```
900 DATA"[illegible]"
901 DATA"[illegible]"
902 DATA"[illegible]"
903 DATA"[illegible]"
904 DATA"[illegible]"
905 DATA"[illegible]"
906 DATA"[illegible]"
907 DATA"[illegible]"
908 DATA"[illegible]"
909 DATA"[illegible]"
910 DATA"[illegible]"
911 DATA"[illegible]"
912 DATA"[illegible]"
913 DATA"[illegible]"
914 DATA"[illegible]"
915 DATA"[illegible]"
916 DATA"[illegible]"
917 DATA"[illegible]"
918 DATA"[illegible]"
919 DATA"[illegible]"
920 DATA"[illegible]"
921 DATA"[illegible]"
922 DATA"[illegible]"
923 DATA"[illegible]"
924 DATA"[illegible]"
925 DATA"[illegible]"
926 DATA"[illegible]"
927 DATA"[illegible]"
928 DATA"[illegible]"
929 DATA"[illegible]"
930 DATA"[illegible]"
931 DATA"[illegible]"
934 DATA"[illegible]"
935 DATA"[illegible]"
936 DATA"[illegible]"
937 DATA"[illegible]"
938 DATA"[illegible]"
939 DATA"[illegible]"
940 DATA"[illegible]"
941 DATA"[illegible]"
942 DATA"[illegible]"
943 DATA"[illegible]"
944 DATA"[illegible]"
945 DATA"[illegible]"
946 DATA"[illegible]"
947 DATA"[illegible]"
948 DATA"[illegible]"
949 DATA"[illegible]"
950 DATA"[illegible]"
951 DATA"[illegible]"
952 DATA"[illegible]"
953 DATA"[illegible]"
954 DATA"[illegible]"
955 DATA"[illegible]"
956 DATA"[illegible]"
957 DATA"[illegible]"
958 DATA"[illegible]"
959 DATA"[illegible]"
960 DATA"[illegible]"
961 DATA"[illegible]"
962 DATA"[illegible]"
963 DATA"[illegible]"
964 DATA"[illegible]"
965 DATA"[illegible]"
966 DATA"[illegible]"
967 DATA"[illegible]"
968 DATA"[illegible]"
969 DATA"[illegible]"
970 DATA"[illegible]"
971 DATA"[illegible]"
972 DATA"[illegible]"
973 DATA"[illegible]"
974 DATA"[illegible]"
975 DATA"[illegible]"
```



ECHT STEREO Raumbilder für beide Augen einzeln

Ein echtes Stereobild ist ein Bild aus zwei Teilen, die für die beiden Augen des Betrachters bestimmt sind und das gleiche Objekt aus den beiden Perspektiven zeigen, die die beiden Augen vor dem echten Objekt hätten; der kleine Unterschied, der auf den Augenabstand zurückgeht, macht es ! In unserem Programm werden einfach alle Punkte in den beiden Teilbildern etwas verschieden stark seitwärts versetzt, und zwar um so mehr, je weiter hinten sie in Wirklichkeit sein sollen, aber in dem einen Bild geht die Differenz nach links, in dem anderen nach rechts. Zum Betrachten nehmen Sie am besten eine Trennwand aus Karton zu Hilfe, damit das rechte Auge nur das rechte Bild und das linke nur das linke zu sehen bekommt. Es erfordert dann etwas Übung, beide Bilder im Gehirn zur Deckung und dennoch scharf zu bekommen. Die Belohnung ist ein echter räumlicher Eindruck. Die Zeile 141 (wenn nicht mit REM unwirksam gemacht) überspringt die Flächen- und Raum-Diagonalen. In Zeile 1ØØ können Sie die beiden Drehwinkel (hier 15 bzw. 40 Grad) durch beliebige andere ersetzen. Statt der Schleife 11Ø-115, die sehr elegant die Koordianten von Würfelecken festlegt, können Sie natürlich auch andere Koordinaten eingeben, entweder mit INPUT oder mit festen Zuweisungen. Die Zeile 141 setzt allerdings voraus, daß die drei Richtungen der Wür-

felkanten den drei Bits in den Zahlen von Ø bis 7 zugeordnet werden (das ist also ein sehr spezieller Trick mit Binärzahlen !). Im übrigen zeichnet das Programm den Würfel in Parallelprojektion (d.h. wie er aus hinreichend großer Entfernung aussieht), nachdem er um zwei Achsen gedreht worden ist.

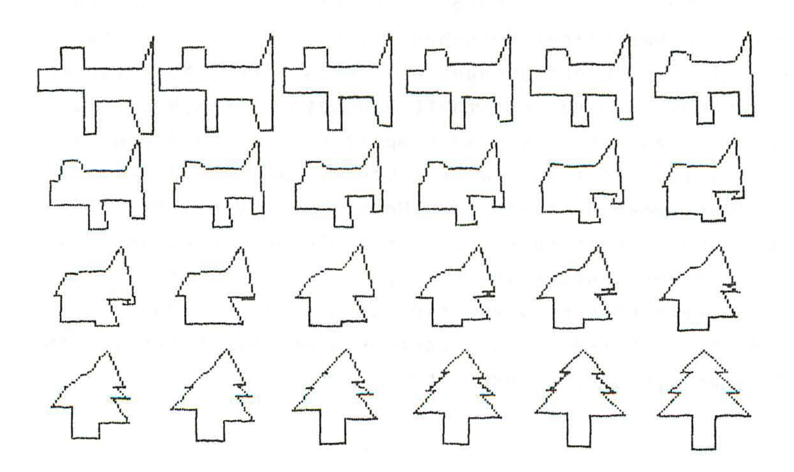
```
90 REM STEREO-BILD WUERFEL
91 :
100 HIRES 1,0:E=2:W1=15*π/180:W2=40*π/180
110 FOR I=0 TO 7
111 X(I)=-1+2*(I AND 4)/4
112 Y(I)=-1+2*(I AND 2)/2
113 Z(I)=-1+2*(I AND 1)
115 NEXT
120 FOR I=0 TO 7
121 U(I)=X(I)*COS(W1)-Y(I)*SIN(W1)
122 V(I)=X(I)*SIN(W1)+Y(I)*COS(W1)
123 W(I)=Z(I)
125 NEXT
130 FOR I=0 TO 7
131 A(I)=W(I)*COS(W2)-U(I)*SIN(W2)
132 B(I)=W(I)*SIN(W2)+U(I)*COS(W2)
133 C(I)=V(I)
135 NEXT
140 FOR I=0 TO 7:FOR J=0 TO I
141 D=ABS((IORJ)-(IANDJ)):IF D<>1 AND D<>2 AND D<>4 THEN 145
143 LINE 80+C(I)*50+B(I)*E,100-A(I)*50,80+C(J)*50+B(J)*E,100-A(J)*50,1
144 LINE 240+C(I)*50-B(I)*E,100-A(I)*50,240+C(J)*50-B(J)*E,100-A(J)*50,1
145 NEXT:NEXT
200 GOTO 200
```

METAMORPHOSEN

Bilder verwandeln sich ineinander

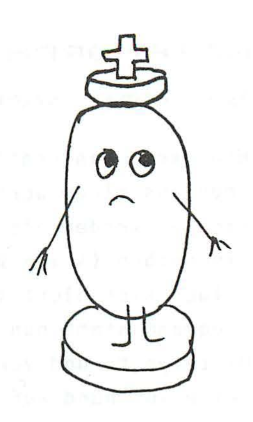
Wie macht man grafisch aus einer Mücke einen Elefanten oder aus einem Schnauzer einen Weihnachtsbaum ? Beide Figuren werden als Umrißlinien mit gleich vielen Ecken eingegeben (X von Ø bis 5Ø, Y von Ø bis etwa 45, damit die Bilder sich nicht ins Gehege kommen). Das relativ einfache Programm nimmt nun einfach von allen Koordinaten gewogene Mittelwerte und verlagert dabei die Gewichtung stufenweise vom Hund auf den Baum.

```
90 REM METAMORPHOSE
91 :
100 INPUT"N ";N:DIM X1(N),X2(N),Y1(N),Y2(N),X0(23),Y0(23)
110 FOR I=1 TO N
115 INPUT"▒X1,Y1 ";X1(I),Y1(I):NEXT
120 FOR I=1 TO N
125 INPUT"▒X2,Y2 ";X2(I),Y2(I):NEXT
130 X1(0)=X1(N):Y1(0)=Y1(N):X2(0)=X2(N):Y2(0)=Y2(N):HIRES 1,0
140 FOR I=0 TO 5:FOR J=0 TO 3
150 X0(I+6*J)=I*53:Y0(I+6*J)=J*50:NEXT:NEXT
160 FOR I=0 TO 23:FOR J=0 TO N-1
170 X1=X0(I)+X2(J)*I/23+X1(J)*(23-I)/23
171 Y1=Y0(I)+Y2(J)*I/23+Y1(J)*(23-I)/23
172 X2=X0(I)+X2(J+1)*I/23+X1(J+1)*(23-I)/23
173 Y2=Y0(I)+Y2(J+1)*I/23+Y1(J+1)*(23-I)/23
175 LINE X1,Y1,X2,Y2,1:NEXT:NEXT
180 GOTO 180
```



WAS MAN SO ALLES SPIELEN KANN

Allgemeines über Spielprogramme



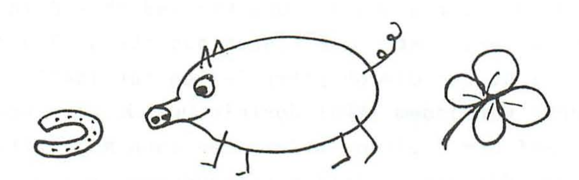
"Spiel" ist hier im Sinne von "Wettbewerb um Punkte" gemeint. Eine oder mehrere Personen spielen; der Computer sagt jedem, was er tun soll oder darf, und führt die Punkte-Konten. Natürlich fragt er zu Beginn nach der Anzahl und den Namen der Spieler und redet später jeden mit seinem Namen an. Am Schluß gibt er eine nach Punkten geordnete Siegerliste aus oder äußert sich sonst über das Verhalten der Teilnehmer. Wenn der Computer nicht nur "die Bank hält", sondern selbst spielt, erfordert das (außer für relativ triviale Spiele wie NIM, besonders einreihiges NIM) sehr pfiffige Programme, die eine schlagkräftige Strategie enthalten. Bei mittelschweren Spielen wie 4-Gewinnt oder Reversi gibt es durchaus BASIC-Programme von vertretbarer Länge, die eine eingebaute Strategie haben. Schachspielende Roboter tauchten bereits im 18. Jahrhundert auf, allerdings als "Fälschung", insofern als sie einen darinsitzenden kleinwüchsigen Menschen (also mit Sicherheit etwas Besseres als ein "Elektronengehirn") enthielten. Noch vor wenigen Jahrzehnten wurde ernsthaft behauptet, Schach sei für Computer viel zu schwer und zu kompliziert (wegen der gewaltigen Verzweigungen schon im Verlaufe von wenigen Zügen). Heute gibt es zwar immer noch ein paar Menschen, die besser als jeder Computer Schach spielen, aber viele verlieren nur deshalb nicht dauernd gegen ihren Computer, weil sie dessen Spielstärke drosseln können. Sie wissen sicher, daß es auch für den C 64 spielstarke Schachprogramme gibt, die natürlich in Maschinensprache geschrieben sind.

Viele Spielprogramme, die Sie in gedruckten Programmsammlungen finden, sind zwar in BASIC, aber nicht für Bildschirmcomputer geschrieben oder nur sehr oberflächlich für ihn umgeschrieben. Sie schreiben meistens das ganze Spielfeld nach jeder Eingabe neu (wie es bei Computern mit Drucker und ohne Bildschirm ja auch nicht anders geht). Sie können das etwas eleganter machen: Nehmen Sie z.B. nur die obersten Zeilen für INPUT und kehren Sie mit HOME vor jedem INPUT dorthin zurück. Die Änderungen der Situation auf dem "Spielbrett" können dann mit PRINT oder POKE ohne Löschen des ganzen Bildes vorgenommen werden.

Was man an einem Tisch (im Gegensatz zu einem Sportplatz) spielen kann, läßt sich auch meistens auf einen Bildschirm-computer übertragen. Es gibt ganze dicke Bücher über die verschiedenen Spiele, und sie geben eine Fülle von Anre-gungen zum Programmieren. Je nachdem, wovon das Gewinnen eines Spieles abhängt, kann man die einzelnen Spiele einer oder mehreren der folgenden Gruppen zuordnen:

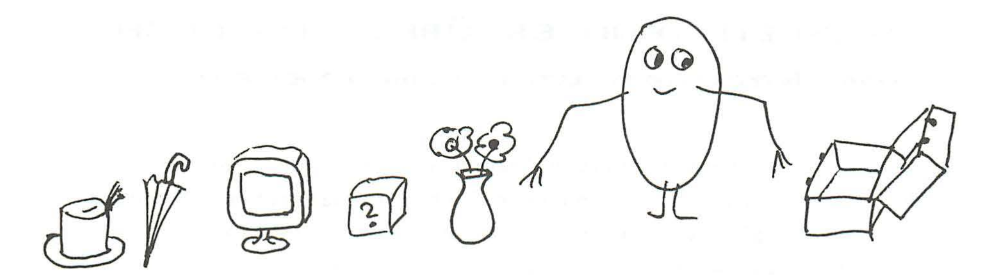
1. Strategiespiele vom einzeiligen NIM (Jeder darf abwech-selnd von einem Haufen 1 bis 3 Steine wegnehmen; wer den letzten nehmen muß, hat verloren) bis zum "königlichen" Schach: Hier kommt es auf vorausschauende Intelligenz an, bei den einfacheren auch ganz entscheidend darauf, wer anfangen darf oder muß. Solche (einfachen) Strategiespiele sollte man so anlegen, daß nach jeder Runde die Aufforderung zum Anfangszug vom Computer richtig adressiert wird und daß die Punktezählung dabei weiterläuft. Spielt der Computer selbst mit, kann er auch mitteilen, ab wann der Mensch durch einen Fehler seinen bis dahin sozusagen sicheren Sieg verschenkt hat (das ist z.B. beim mehrzeiligen NIM möglich, bei dem beliebig viele Steinchen, aber mehr als Ø, aus genau einer Zeile entnommen werden müssen; ob der 1. oder der 2. Spieler bei fehlerfreiem Spiel gewinnt, hängt von der Anfangs-verteilung der Steinchen auf die Zeilen ab).

Das Programmieren von Strategiespielen ist im allgemeinen kompliziert; selbst die Übertragung der Spielregeln in ein Programm, das nur das Spiel mehrerer Menschen organisiert, überwacht und bewertet (und natürlich auf den Bildschirm bringt), ist (z.B. schon für Mühle) nicht ganz einfach.



2. <u>Glücksspiele</u>. Während es bei Strategiespielen nur auf die Intelligenz ankommt, spielt sie hier gar keine Rolle (bei den "reinen"Glücksspielen). Es ist klar, daß der Zufållsgenerator RND(Ø) hier die entscheidende Rolle spielt. Denken Sie bitte daran, daß RND(Ø) mindestens zu Beginn eines solchen Programms vorkommen muß: RND (1) erzeugt nach jedem Einschalten des Gerätes nacheinander die gleichen "Zufallszahlen". Das ist natürlich äußerst unfair, weil der Kenner des Programms sich die Reihenfolge der gewürfelten Zahlen dann bald merken kann. Die Programme sind im Prinzip extrem einfach; man kann natürlich sehr viele Zutaten bringen (Kontoführung, Roulette-Tisch mit Bezifferung, akustisches Rollenlassen der Kugel, Anzeige der durchlaufenden Zahlen bei einem "einarmigen Banditen" usw.). Sicher sind reine Glücksspiele langweilig, wenn es <u>nicht</u> um Geld geht, und gefährlich, <u>wenn</u> es darum geht. Man kann aber auch an ihnen demonstrieren, daß Spielsysteme nicht funktionieren (z.B. das System, auf Rot den kleinsten Einsatz zu setzen, den Einsatz bei jedem Verlust zu verdoppeln und bei jedem Gewinn wieder ganz klein anzufangen). Unter uns: Wenn ich ein System hätte, würde ich kein Buch darüber schreiben, sondern die Spielbanken sprengen.

3. Gedächtnisspiele. Hier sind Zahlen, Bildsymbole, Wörter oder Melodien, die der Computer vorführt, nach einer gewissen Zeit wiederzuerkennen oder zu benennen und mit den Tasten anzugeben. Natürlich kann bei Gedächtnisspielen der Zufall, der gelegentlich ein besseres Gedächtnis vortäuscht, nie ganz ausgeschlossen werden; das gleicht sich aber nach längeren Spielfolgen aus. Ein Spiel, das gleichermaßen bei der Aufstellung des Programms wie beim Spielen Spaß macht, ist ein Bilder-Memory: Vorbestimmte Bilder sind unsichtbar und zufällig über den Bildschirm verteilt (z.B. im Raster 7 x 7 Bilder aus je 3 x 3 Buchstabenfeldern). Die Spieler werden der Reihe nach aufgefordert, zwei Bilder (durch Anwahl der Rasterzahlen mit GET) "aufzuklappen". Der Computer zeigt sie dann an den entsprechenden Stellen. Wer zwei gleiche Bilder (denn fast jedes Bild kommt zweimal vor) findet, bekommt einen Punkt und eine akustische Belohnung. Am Schluß zeigt der Computer die Rangfolge der Spieler, die abgelaufene Zeit und die Zahl der benötigten Aufklappvorgänge an.



Eine andere Form des Gedächtnisspieles ist als Kinderspiel "Ich pack' in meinen Koffer: ..." und als Spielgerät "Senso" bekannt: Eine Kette aus zuerst nur einem Wort, einem Ton, einer Farbe oder einer Ziffer wird von Mal zu Mal um ein Element länger (natürlich per Zufallsgenerator); der Spieler muß jedesmal die ganze Folge aus dem Gedächtnis eintippen.

Die Gedächtnisspiele sind nicht nur ein gutes Training für die Konzentration, sondern gestatten auch dem Programmierer, seine ganze Phantasie in die Auswahl und Gestaltung von

Wörtern oder Bildern zu legen, nicht nur bei eigenen Programmen, sondern auch bei anderen, wenn die LIST zugänglich ist. Noch ein Tip: Seien Sie nicht traurig, wenn Sie gegen kleine Kinder verlieren: die können das wirklich besser !

4.Geschicklichkeitsspiele. Hier geht es meist darum, rechtzeitig die richtige Taste (oder den Joystick in die beste Richtung) zu drücken. Reaktionsgeschwindigkeit ist dabei meist sehr wichtig, oft muß man aber auch gewisse Bewegungsabläufe abschätzen und einkalkulieren. In kommerziellen Spielen sieht das meist sehr kriegerisch aus, aber ein schnell laufender gefräßiger Wurm, der durch erfolgreiches Fressen immer länger wird, oder ein Autorennen, bei dem physikalische Gesetze zu beachten sind, sind auf lange Sicht meist mindestens ebenso spannend. Praktisch alle Spiele in den elektronischen Spielhöllen sind in diesem Sinne Geschicklichkeitsspiele (ein C 64 amortisiert sich da sehr schnell).

RECHNEN KANN ER ÜBRIGENS AUCH

Vom Rechnen und Simulieren

Für viele wissenschaftliche Themen wird man den C 64 als Hilfsmittel zur Ausführung numerisch-mathematischer Methoden verwenden, also von Rechnungen, die entweder in kleinen Schritten vorgehen oder stufenweise immer bessere Genauigkeiten erreichen. Viele Anleitungen dazu finden Sie in den Programmsammlungen, die es für BASIC (ohne Bindung an bestimmte Computertypen) und für programmierbare Taschenrechner gibt. Vergessen Sie dabei aber nicht, die Dialogfähigkeit und die Möglichkeiten zur grafischen Ergebnisausgabe zu nutzen, wie sie Bildschirmcomputer im allgemeinen und der C 64 in besonders hohem Maße hat (sonst mißbrauchen Sie den C 64 fast als programmierbaren Taschenrechner, nutzen ihn jedenfalls nicht angemessen aus !).

Besonders günstig sind Simulationsprogramme, bei denen ein zeitlicher Vorgang aus den Bereichen der Natur-, Sozial- oder Wirtschaftswissenschaften so behandelt wird, daß wichtige Kenngrößen in der richtigen zeitlichen Reihenfolge aufeinander aufbauend berechnet werden, gegebenenfalls mit Berücksichtigung von Eingaben, die man während des Programmlaufs machen kann (Kraft- oder Spannungsstöße in der Physik, Zugaben von Reagenzien in der Chemie, Aktienkäufe bei einer Börsen-Simulation, Änderung der Geburtenziffer bei Bevölkerungspyramiden usw.). Anregungen hierzu erhalten Sie am besten bei der Lektüre von Lehrbüchern der einzelnen Wissenschaften: Benutzen Sie Ihren Computer, um grafische Darstellungen zum Leben zu erwecken !

Einige Beispiele folgen ab Seite 219.

Wie genau ist Ihr C 64 eigentlich ?

Wenn es sein muß, unendlich genau. Nicht etwa nur bei den Aufgaben, die glatt aufgehen. Mit "unendlich" ist natürlich (wie sonst auch) gemeint: Es gibt keine Grenze, die nicht überschritten werden könnte. Das folgende kleine Programm führt eine relativ einfache Rechnung* beliebig genau aus, wenn Sie zwei ganze Zahlen eingeben:

```
10 INPUT Z,N
20 IF N=<Z THEN 40
30 PRINT CHR$(48+B);:Z=Z*10:B=0:GOTO 20
40 Z=Z-N:B=B+1:GOTO 20
```

In dieser einfachen Form zeigt es allerdings keinen Dezimalpunkt an. Die 10 in Zeile 30 können Sie auch gegen eine andere Ziffernsystembasis austauschen; die ASCII-Zeichen in Zeile 30 sind natürlich die Ziffern; für Basen größer als 10 müßte man hier noch Ergänzungen einbauen.

---

* *Welche könnte das sein ?*

FIBONACCI UND DER GOLDENE SCHNITT

Der größte Mathematiker des Mittelalters hieß Leonardo Pisano (d.h. aus Pisa), er wurde auch Fibonacci (gesprochen: fibonattschi) nach seinem Vater Bonacci genannt und lebte um 1200. Nach ihm ist eine Zahlenfolge benannt, die mit Ø und 1 anfängt und in der dann jede neue Zahl gleich der Summe ihrer beiden Vorgänger ist, also:

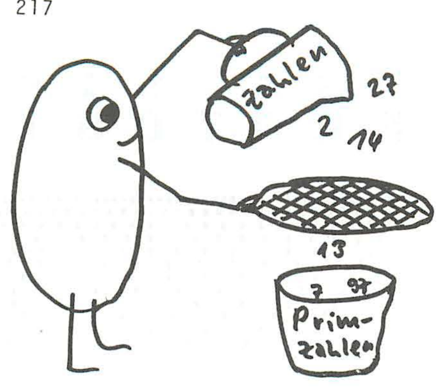
```
10 A=Ø:B=1
20 PRINT A:IF B<>Ø THEN PRINT TAB(9)A/B
30 C=A+B:A=B:B=C:GOTO20
```

Sie können auch A und B beliebig starten: 1Ø INPUT A,B (nur zwei Nullen sind wenig sinnvoll). Der Quotient A/B strebt recht schnell gegen eine Konstante Ø.618Ø33989...=G die als Goldener Schnitt bezeichnet wird, da für sie gilt: (1-G)/G=G/1. Daß unsere Fibonacci-Folge gegen diesen Wert konvergiert, ist leicht einzusehen, wenn man als gegeben hinnimmt, daß sie überhaupt konvergiert. Das gilt übrigens unabhängig von A und B (sofern nicht beide gleich Ø sind), auch negative A oder B sind zugelassen. Für den Fall, daß die Ø in der Folge vorkommt, ist jede Differenzenfolge (beliebig hoher Ordnung) mit der Folge selbst identisch (aber natürlich etwas versetzt). (Eine Differenzenfolge entsteht, wenn man neben eine Folge die Differenzen der benachbarten Glieder schreibt; macht man das mit der Differenzenfolge wieder, so kommt man zur nächsthöheren Ordnung.)

Als nicht sehr ernst gemeintes Beispiel dafür, daß man auch manchmal Programme, die nicht völlig trivial sind, in eine einzige Zeile quetschen kann, dasselbe etwas kürzer:

```
1Ø PRINT X+1;TAB(9)(X+1)/(Y+2):Z=X+1:X=Y+1:Y=Y+Z:GOTO 1Ø
```

Da die Anfangswerte hier zwangsläufig beide Ø sind, ist die Umbenennung in X+1 und Y+1 nötig.

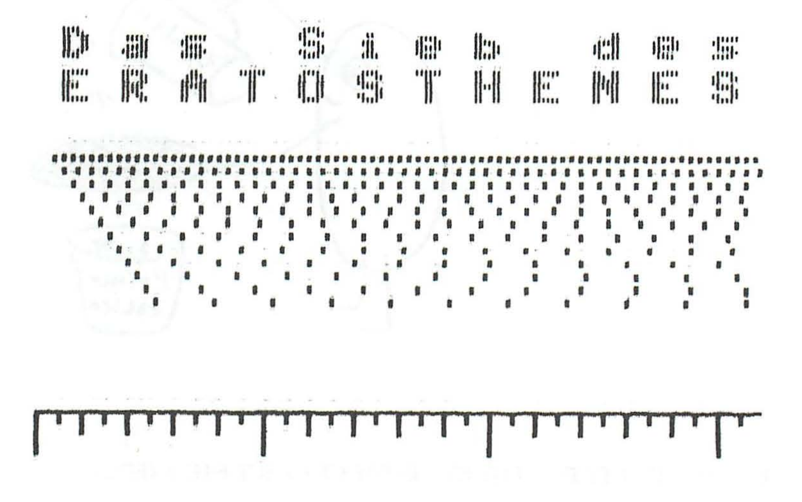


DAS SIEB DES ERATOSTHENES

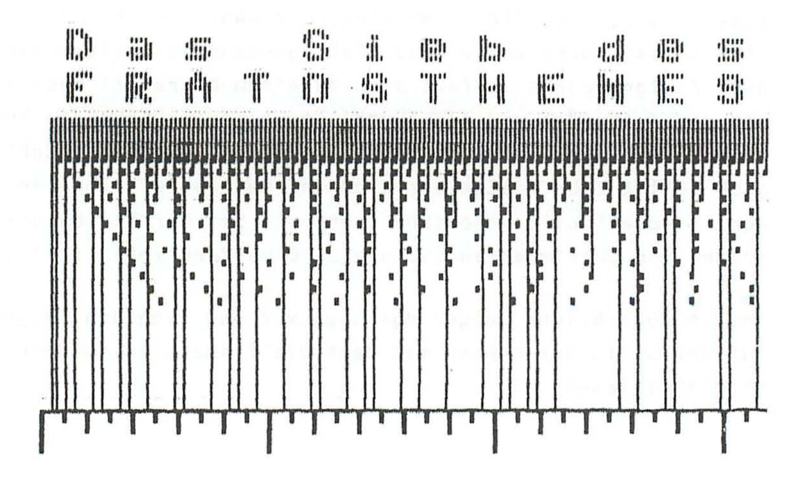
Primzahlensuche ganz anschaulich

Eratosthenes aus Kyrene (in Afrika) lebte fast im ganzen 3. vorchristlichen Jahrhundert, er war Direktor der berühmten Bibliothek von Alexandria und bestimmte den Erdumfang. Seine Methode, Primzahlen zu suchen, können wir auf dem Bildschirm besonders anschaulich vorführen. Wir beschränken uns wegen der MULTIcolor-Auflösung auf die Zahlen von Ø bis 159 und markieren darin alle ganzzahlig-Vielfachen von allen Zahlen zwischen 2 und der Quadratwurzel aus 159 (oder mehr). "Viel" bedeutet darin "mehr als Eins". Wenn es nun für alle X-Werte von oben herab "regnet", fangen uns diese Marken die Vielfachen der genannten Zahlen ab; was durch das Sieb durchgelassen wird, sind die Primzahlen, die außer durch Eins und sich selbst durch keine ganze Zahl ganzzahlig teilbar sind. Die "Zahlentheorie" befaßt sich zu einem beträchtlichen Teil mit dem Auffinden und den Verteilungseigenschaften der Primzahlen, auf die man statistische Gesetze anwendet, obwohl sie doch völlig klar festgelegt sind und keinem Zufall unterliegen. Eine aktuelle Bedeutung haben sie bei der Verschlüsselung von Geheimzahlen (Spionage, Computerkriminalität usw.).

Unsere COPY-Bilder zeigen das Sieb vor und nach dem "Regnen". Auf der Skala unten kann man dann die Primzahlen im Bereich bis 159 ablesen.

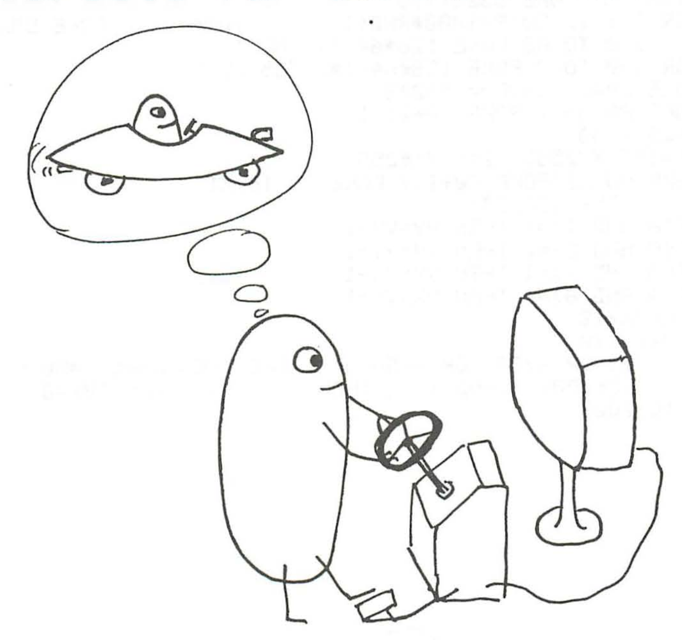



```
90 REM  DAS SIEB DES ERATOSTHENES - GANZ ANSCHAULICH
91 :
100 HIRES 1,15:MULTI 10,6,6:POKE 53280,15
110 TEXT 5,3,"█▔AS ♦IEB DES",2,2,13
111 TEXT 5,23,"ERATOSTHENES",2,2,13
120 FOR Y=2 TO 13:FOR X=2*Y TO 159 STEP Y
130 LINE X,6*Y+50,X,6*Y+52,3:NEXT:NEXT
131 LINE 0,180,159,180,3
132 FOR X=0 TO 155 STEP 5:LINE X,181,X,185,3:NEXT
133 FOR X=0 TO 150 STEP 10:LINE X,181,X,190,3:NEXT
134 FOR X=0 TO 150 STEP 50:LINE X,181,X,199,3:NEXT:COPY
140 FOR X=2 TO 159:Y=45
150 IF TEST(X,Y)<>3 THEN PLOT X,Y,1:Y=Y+1:GOTO 150
160 NEXT:COPY
500 GOTO 500
```



NICHTS FÜR SIMULANTEN

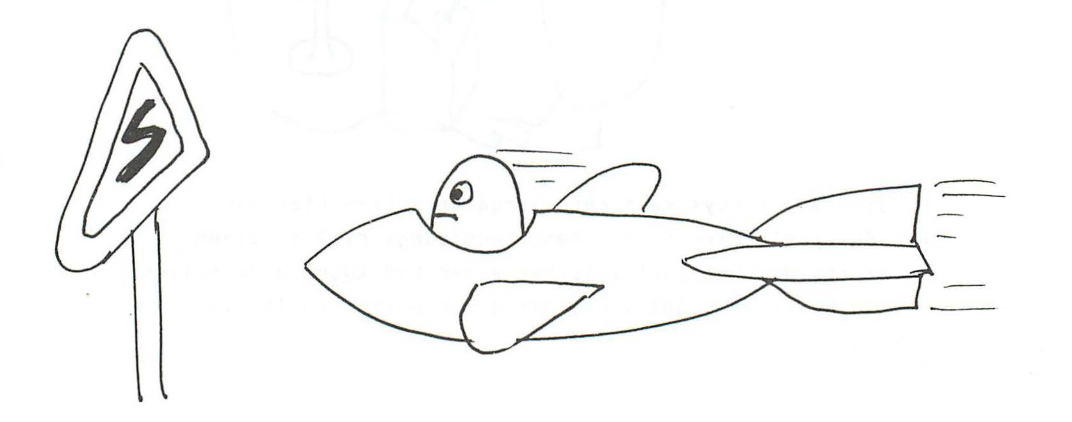
Beispiele für Simulationen



Zur Simulation physikalischer Vorgänge sollen hier nur drei Beispiele dienen: ein Beschleunigungsspiel in einem Labyrinth, der Wettlauf zwischen Besen und Kugel beim Fallen und ein Heiz- und Kühlsystem mit einem einfachen Thermostaten. -

Das Beschleunigungsspiel ist ein Hindernisrennen um einige zufällig herumliegende Steinbrocken. Mit dem Joystick an Port 2 geben Sie in jedem Augenblick die Beschleunigungsrichtung (bzw. den unbeschleunigten Zustand) ein. Der Start des roten "Raumschiffs" ist links oben, als Ziel können Sie die rechte untere Ecke betrachten. Sie können aber auch ziellos einen Slalom fahren. Bei jedem Zusammenstoß mit einem Stein oder dem Rand werden Sie unerbittlich

```
100 REM HINDERNISRENNEN MIT BESCHLEUNIGUNG
101:
102 REM JOYSTICK AN PORT 2
103:
105 PRINT"":POKE 53281,15
110 FOR I=1 TO 30:P=1000*RND(1):POKE 1024+P,81:POKE 55296+P,6:NEXT
115 FOR I=0 TO 63:POKE 128*64+I,0:NEXT
120 FOR I=0 TO 7:POKE 128*64+I*3,255:NEXT
130 POKE 2040,128:P0= 53248
140 POKE P0+39,2:POKE P0+21,1
150 X=25:Y=50
200 X1=INT(X/255):X2=X-X1*255
210 POKE P0,X2:POKE P0+1,Y:POKE P0+16,X1
220 A=127-PEEK(56320)
221 IF(A AND 1)=1 THEN VY=VY-1
222 IF(A AND 2)=2 THEN VY=VY+1
223 IF(A AND 4)=4 THEN VX=VX-1
224 IF(A AND 8)=8 THEN VX=VX+1
230 X=X+VX/10
240 Y=Y+VY/10
250 IF X<25 OR X>335 OR Y<50 OR Y>242 THEN X=25:Y=50:VX=0:VY=0
260 IF (PEEK(P0+31)AND 1)=1 THEN X=25:Y=50:VX=0:VY=0
270 GOTO 200
```

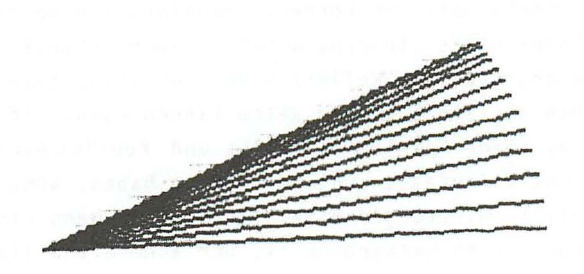


an den Start zurückversetzt. Die Zeile 110 mit den zufälligen Steinen (deren Zahl nicht unbedingt 30 sein muß) können Sie durch eine Folge von PRINT-Anweisungen ersetzen, die Ihnen ein schönes Labyrinth malt. Das Raumschiff ist hier ein schlichtes Quadrat; gestalten Sie es schöner, verzieren Sie es (mit weiteren Sprites z.B.) während der Beschleunigungsphasen mit ausgestoßenen Gasstrahlen ! Sie

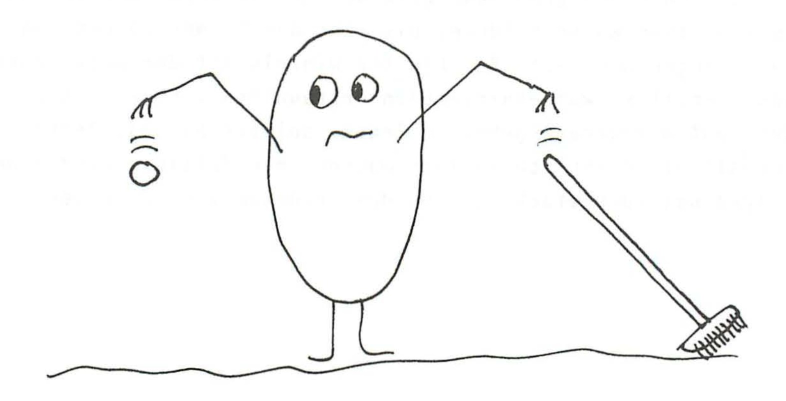
können auch eine ganze Folge von Raumschiffen, die in verschiedene Richtungen fahren, erzeugen und in Abhängigkeit von der tatsächlichen Fahrtrichtung als Sprite aufrufen. Sie können das Spiel auch auf zwei Personen ausweiten, da wir ja zwei Joysticks abfragen können; normalerweise werden Sie dann zwei Raumschiffe steuern, wobei es auch interessant sein kann, eine gegenseitige Kollision der Fahrzeuge zuzulassen, damit man ungestört um die Wette fahren kann; eine andere Idee wäre, beide Spieler als Pilot und Kopilot eines einzigen Fahrzeugs aufzufassen, die Mißerfolg haben, wenn sie widersprüchliche Richtungen einschlagen, etwa wenn einer ein Hindernis von rechts umfahren will, Der andere von links. Zeile 22Ø fragt den Joystick an Port 2 ab, die folgenden 4 Zeilen werten die Abfrage als Beschleunigungskomponenten, 23Ø bis 24Ø übertragen die Geschwindigkeitskomponenten auf die Ortskoordinaten, 25Ø und 26Ø fragen auf Kollision mit dem Rand und mit den Hindernissen ab.

Das zweite Simulationsbeispiel handelt von einem Besen(-stiel), der schneller umfällt als eine aus der gleichen Höhe fallende Kugel für den Fall braucht. Nach der Eingabe des Startwinkels (gegen den Boden gemessen) zeichnet das Programm den Vorgang in einzelnen Momentbildern, bis eins der beiden Objekte am Boden angekommen ist. Bei kleinen Winkeln ist der Besen durchaus schneller, was einigermaßen erstaunlich ist. Wenn Sie Wert auf genauere Ergebnisse legen, sollten Sie das Zeitintervall DT wesentlich kleiner machen; die Zeitlupe wird dann allerdings auch stärker, d.h. das Programm dauert länger.

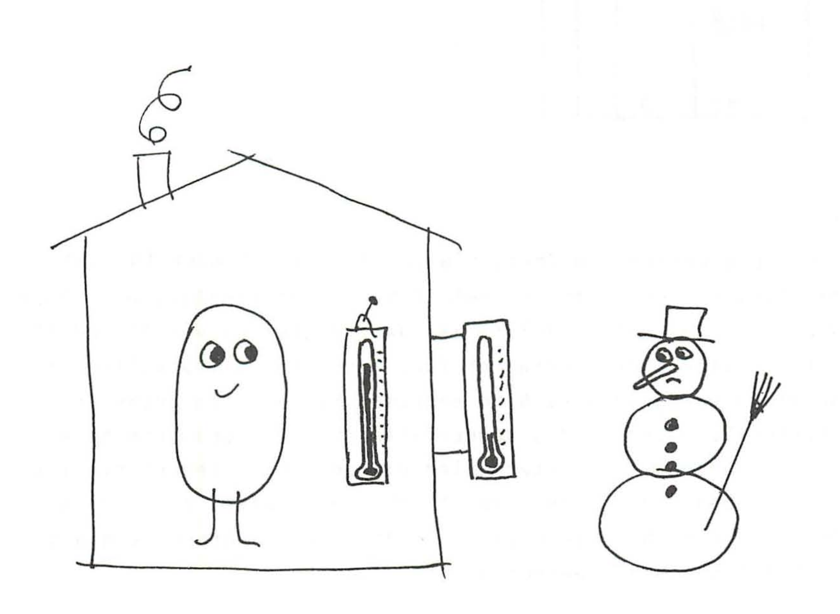
```
Der Besen ist unten !
                      Der Startwinkel
                         war  30 Grad.
```



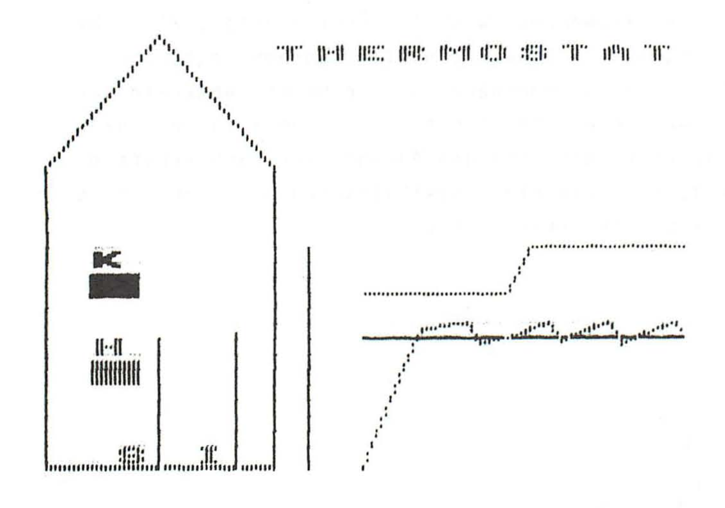
```
1000 INPUT"WINKEL     30████";WS:W=WS*π/180
1010 H=SIN(W):L=1:MA=1:G=-9.8:DT=.02
1100 HIRES 1,0
1210 PLOT 300,199-H*190,1
1220 LINE 30,199,30+222*COS(W),199-190*SIN(W),1
1230 DM=COS(W)*L/2*MA*G
1240 WB=DM/MA/L↑2*3
1250 WG=WG+WB*DT
1260 W=W+WG*DT
1270 V=V+G*DT
1280 H=H+V*DT
1290 IF W>0 AND H>0 GOTO 1210
1295 IF W>0 THEN TEXT03,5,"▌▛IE ╱UGEL IST UNTEN !",1,1,9
1296 IF H>0 THEN TEXT03,5,"▌▛ER |ESEN IST UNTEN !",1,1,9
1297 TEXT 140,18,"▌▛ER ♦TARTWINKEL ",1,1,9
1298 TEXT 168,30,"▌█WAR "+STR$(WS)+" |RAD.",1,1,9
1300 GOTO 1300
```



Zum Verständnis des Programms muß man mit Begriffen wie Drehmoment, Trägheitsmoment, Winkelgeschwindigkeit, Winkelbeschleunigung usw. bekannt sein. Im Programm erscheinen nur die einfachsten Zusammenhänge dieser Größen untereinander (insbesondere ihre Definitonen); trotzdem zeigt uns das Programm ganz klar, daß ohne die Annahme von Zauberkräften allein aus diesen bekannten physikalischen Gesetzen das seltsame Verhalten zu erklären ist.



Das Programm "Thermostat" zeigt ein Haus mit drei Temperaturskalen: innen für den Sollwert (S) und den Istwert (I), außen für die Störgröße "Außentemperatur". Im Programmtext heißen sie TS, TH und TA (TI ist nicht zulässig !). Ein Heizgerät H und ein Kühlgerät K schalten sich ein und aus, wenn gewisse Abweichungen der Ist- von der Solltemperatur über-



schritten werden; im übrigen wirkt die Außentemperatur auf den Istwert innen ein (Wärmeleitung). Das Einzige, was Ihnen außer dem Zuschauen bleibt, ist das Verstellen von TA und TS mit den vier Funktionstasten (Dauerfunktion !). Schaffen die Geräte die Regelung auch in extremen Fällen ? Im größeren rechten Teil des Bildes werden alle drei Temperaturen höhengleich zu den Thermometersäulen wie in einer Fieberkurve mitgeschrieben. Das Ganze scheint sehr kompliziert zu sein, das Programm ist aber (jedenfalls in der hier gebotenen einfachen Fassung) durchaus überschaubar:

1ØØ-126 Zeichnung
16Ø-17Ø Anfangswerte
2ØØ-2Ø4 Abfrage der Funktionstasten für Änderungen von TS und TA
21Ø-242 Auftragung der Temperaturen als Thermometersäulen und als Punkte im Zeitdiagramm

250 Wärmeleitung (so einfach geht das !)
260-265 Wirkung von Heizung und Kühlung
270-273 Ein- und Ausschalten von H und K bei den Schaltpunkten
280-281 Anzeigen von H und K im Bild (Variante: statt 1+H bzw. 1+2*K schreiben: 2*H bzw. 3*K, dann erscheinen die ausgeschalteten Geräte schwarz)
290-300 Fortschreiten auf Zeitskala und spaltenweises Löschen
310 Rücksprung zum Schleifenanfang

```
90 REM THERMOSTAT
91 :
92 REM MIT SIMON'S BASIC
93 :
100 POKE53280,15:HIRES 0,1:MULTI 11,10,14
110 TEXT 50,5,"THERMOSTAT",1,1,9
111 BLOCK 10,150,20,160,1:CHAR 10,140,8,1,1
112 BLOCK 10,110,20,120,1:CHAR 10,100,11,1,1
113 REM   GOTO 17,1,"S I"
114 CHAR 15,190,19,1,1:CHAR 32,190,9,1,1
121 LINE 0,199,50,199,1
122 LINE 0,198,50,198,1
123 LINE 0,60,0,199,1
124 LINE 50,60,50,199,1
125 LINE 0,60,25,0,1
126 LINE 50,60,25,0,1
160 T=70:T1=80
170 TA=80:TS=60
200 E=PEEK(203)
201 IF E=4 AND TA<140 THEN TA=TA+2
202 IF E=5 AND TA>  3 THEN TA=TA-2
203 IF E=6 AND TS<140 THEN TS=TS+2
204 IF E=3 AND TS>  3 THEN TS=TS-2
210 LINE 25,60,25,199-TS,0:LINE 25,199-TS,25,199,2
220 LINE 42,60,42,199-TH,0:LINE 42,199-TH,42,199,2
230 LINE 58,60,58,199-TA,0:LINE 58,199-TA,58,199,2
240 PLOT T,199-TS,3
241 PLOT T,199-TH,2
242 PLOT T,199-TA,1
250 TH=TH+(TA-TH)/100
260 IF H=1 THEN TH=TH+2
265 IF K=1 THEN TH=TH-2
270 IF TH<TS-2 THEN K=0
271 IF TH<TS-6 THEN H=1
272 IF TH>TS+2 THEN H=0
273 IF TH>TS+6 THEN K=1
280 BLOCK 10,150,20,160,1+H  :CHAR 10,140,8,1+H,1
281 BLOCK 10,110,20,120,1+2*K:CHAR 10,100,11,1+2*K,1
290 T=T+.5:T1=T1+.5
291 IF T >159 THEN T =65
292 IF T1>159 THEN T1=65
300 LINE T1,60,T1,199,0
310 GOTO 200

READY.
```

EIN PROGRAMM SCHREIBT SICH (FAST) SELBST

Wenn sich in einem BASIC-Programm Bilder schnell bewegen sollen, empfiehlt sich PRINT. Wir wollen einen kleinen Trickfilm einer ("stehenden" oder laufenden) Welle erzeugen, die mit den waagerechten Strichen der Tastaturgrafik gezeichnet wird, und zwar aus 50 Momentbildern, die zyklisch nacheinander in die gleiche Zeile geschrieben werden. Die Arbeit, die zugehörigen Codenummern zu berechnen und einzutippen, ist so recht für einen Intellekt, der fehlerfrei Kompliziertes rechnen kann, ohne sich über Stumpfsinn zu ärgern, für den Computer also zum Beispiel. Wir geben ihm zunächst 30 gleichlautende Zeilen ein (natürlich mit Überschreiben der Zeilennummer und Return-Taste !):

```
100 PRINT"1234567890123456789012345678901234567890]";
101 PRINT"1234567890123456789012345678901234567890]";
102 PRINT"1234567890123456789012345678901234567890]";
```

usw. bis

```
128 PRINT"1234567890123456789012345678901234567890]";
129 PRINT"1234567890123456789012345678901234567890]";
130 GOTO 100
```

Dieses Programm für sich alleine druckt immer wieder den gleichen Text an die gleiche Stelle. Mit der direkten Anweisung FOR I=0 TO 100:PRINT PEEK(2048+I);:NEXT sehen Sie, wie die ersten beiden Zeilen im Speicher stehen: Bei 2049 beginnt das BASIC-Programm im C 64, und zwar kommen zunächst zwei Bytes, die anzeigen, bei welchem Speicherplatz die nächste BASIC-Zeile beginnt; die Zahl ist das 256-fache des zweiten Bytes plus dem ersten Byte. Die nächsten Bytes sind die in gleicher Weise verschlüsselte

Zeilennummer. Es beginnt dann die eigentliche Zeile mit PRINT (Code 153) und Gänsefüßchen (34), dann kommen unsere 40 Zifern. Der Schluß der Zeile wird mit der 0 angezeigt. Die ganze BASIC-Zeile ist also 5 Bytes länger als die Zahl ihrer Schlüsselwörter und Klartextzeichen vorgibt, in unserem Fall also 50. Der Sinn der Sprungadressen besteht übrigens darin, daß bei Sprüngen im Programm nicht der ganze Text, sondern nur diese Adressen gelesen werden müssen: der Interpreter hangelt sich so sehr schnell durch das Programm.

Nun geben wir den zweiten Teil des Programms dazu:

```
200 DATA 175,210,198,192,195,196,197,183
210 FOR I=0 TO 7:READ A(I):NEXT
220 FOR T=0 TO 29:FOR X=0 TO 39
230 Y=3.5+3.5*COS(T*π/15-X*π/10)
240 POKE 2055+50*T+X,A(Y):NEXT:PRINT T:NEXT
250 PRINT"□"
260 GOTO 100
```

Es POKEt neue Zeichen an die Stelle der willkürlichen (aber genau abgezählten !) Ziffern, und zwar berechnet es erst einmal die orts- und zeitabhängigen Funktionswerte in Zeile 230 (in einem Bereich von 0 bis 7). Anhand der Zuordnungsfunktion A(I) bekommt dann jede Zahl als Grafikzeichen einen Strich der entsprechenden Höhenlage (im ASCII-Code). Es ist sehr wichtig, daß diese Zeichen genau an die richtigen Stellen (und vor allem nicht in die Sprungadressen !) gePOKEt werden, sonst hängt sich das Programm beim Starten auf.

Beim ersten Mal startet man mit RUN 200. Es erscheinen die Zahlen von 0 bis 29 und berichten vom Fortgang der Berechnungen Dann erscheint die laufende Welle. LISTen Sie nun das Programm aus, sehen Sie viele gegeneinander versetzte Wellenbilder in den Zeilen 100 bis 129. Neu können Sie mit RUN starten; die Berechnungen sind ja jetzt nicht nötig, das Programm hat sich selbst in die interessante

Form gebracht und Ihnen viel Arbeit abgenommen (es ist noch gar nicht so viele Jahre her, daß ich erst ein solches Programm "zu Fuß" geschrieben habe und leider erst dann auf die Idee kam, wie man die Arbeit auf den Computer abwälzen kann).

Ändern Sie das Vorzeichen vor dem X in Zeile 23Ø, so läuft nach einem neuen Rechendurchgang (d.h. RUN 2ØØ) die Welle in der umgekehrten Richtung. Stehende Wellen bekommen Sie mit der Variante: 23Ø Y=3.5+3.5*COS(T*π/15)*COS(X*π/1Ø).

Natürlich können Sie auch die Wellenlänge ändern, indem Sie den Faktor hinter dem X ändern. Jede durch Neuberechnung erstellte Version des Programms können Sie natürlich auch SAVEn und dann vorführen, ohne jedesmal die Rechenzeit zu benötigen.



A N H A N G

EIN ÜBERBLICK ÜBER DAS BASIC DES C 64

BASIC-Schlüsselwörter werden im Speicher als jeweils ein Byte festgehalten , wobei die Nummer stets größer als 127 ist; das zugehörige Zeichen im Bildschirmcode erscheint revers. (mit FOR I=Ø TO 1ØØ:A=PEEK(2Ø47+I):PRINT A;:NEXT können Sie die Bytes als Zahlen ausdrucken
in komprimierter Schreibweise mit:

```
FOR I=Ø TO 3ØØ:A=PEEK(2Ø48+I):POKE 1Ø24+I,A:
POKE 55296+I,6:NEXT
```

wobei Sie zum besseren Lesen auf den zweiten Zeichensatz umschalten sollten (SHIFT & C= drücken).

Beim Eingeben eines BASIC-Programms brauchen Sie nicht unbedingt die ganzen Schlüsselwörter zu tippen; stets genügen maximal 3 Buchstaben, meist ist der letzte mit SHIFT zu tippen. In den folgenden Listen wird angenommen, daß Sie dabei den zweiten Zeichensatz verwenden (Klein- und Großbuchstaben).

1. <u>Funktionen mit Dummy-Argumenten</u>, z.B. (Ø)

Die Spalten bedeuten der Reihe nach:
a) normale Bezeichnung, b) kürzeste Form bei der Eingabe (hier im zweiten Zeichensatz dargestellt), c) Code-Byte
d) Bildschirmzeichen zum Code-Byte, revers zu verstehen,
e) kurzer Hinweis auf die Bedeutung

a	b	c	d	e
FRE	fR	184	8	freie Bytes für BASIC
POS	pos	185	9	aktuelle Spaltennummer

Anmerkung: pos ist ein Beispiel für Schlüsselwörter ohne Abkürzung, poS ist nicht möglich. - Natürlich brauchen Sie zur Benutzung der Abkürzungen nicht den 2. Zeichensatz zu nehmen, auf dem Bildschirm erscheinen dann Großbuchstaben und Grafikzeichen, z.B. N¯ statt nE für NEXT.

2. Funktionen von Zahlen, die selbst Zahlen liefern

ABS	aB	182	6	Absolutbetrag
ATN	atn	193	A	Arcustangens (im Bogenmaß)
COS	cos	19Ø	>	Cosinus (vom Bogenmaß)
EXP	eX	189	=	Exponentialfunktion zur Basis e
INT	int	181	5	nächste nichtgrößere ganze Zahl
LOG	log	188	<	Logarithmus zur Basis e
PEEK	pE	194	B	Inhalt des Speicherplatzes
RND	rN	187	;	Zufallszahl: RND(Ø) zeitabhängig,
				RND(1) nur reihenfolgenabhängig
SGN	sG	18Ø	4	Vorzeichen: -1,Ø,1
SIN	sI	191	?	Sinus (vom Bogenmaß)
SQR	sQ	186	:	Quadratwurzel
TAN	tan	192	_	Tangens (vom Bogenmaß)
USR	uS	183	7	Maschinen-Unterprogramm als Funktion

3. Stringfunktionen von Strings und Zahlen

LEFT$	leF	2ØØ	H	LEFT$("ABCDEF",2) ist "AB"
MID$	mI	2Ø2	J	MID$("ABCDEF",4,2) ist "DE"
				MID$("ABCDEF",2) ist "BCDEF"
RIGHT$	rI	2Ø1	I	RIGHT$("ABCDEF",2) ist "EF"

4. Stringfunktionen von Zahlen

CHR$	cH	199	G	Zeichen zum ASCII-Code
STR$	stR	196	D	Zahl als Zeichenkette

5. Zahlenfunktionen von Strings

ASC	aS	198	F	ASCII-Codenummer zum Zeichen
LEN	len	195	C	Länge einer Zeichenkette
VAL	vA	197	E	(Anfang einer) Zeichenkette als Zahl

6. Anweisungen und Verknüpfungen:

+	+	17Ø	*	Addition der Ausdrücke davor und danach
-	-	171	+	Subtraktion (entspr.) oder Vorzeichen
*	*	172	,	Multiplikation
/	/	173	-	Division

↑	↑	174	.	Potenzierung
>	>	177	1	Vergleich: ob links größer als rechts
=	=	178	2	Vergleich: ob links und rechts gleich
<	<	179	3	Vergleich: ob links kleiner als rechts
AND	aN	175	/	logische Konjunktion (UND)
CLOSE	clO	160	□	Schließen eines Files
CLR	cL	156	£	Löschen der Variablen
CMD	cM	157	]	lenkt PRINT auf externe Geräte um
CONT	cO	154	x	abgebrochenen Lauf fortsetzen
DATA	dA	131	c	die folgenden Zahlen oder Strings gehören zum Datensatz (für READ)
DEF	dE	150	v	es folgt die Definition von FN ...
DIM	dI	134	f	Platzreservierungen für indizierte Variable
END	eN	128	@	Ende des Laufs (ohne Meldung der Zeile)
FN	fn	165	%	Anfang der Bezeichnung einer selbstdefinierten Funktion, z.B. FNY
FOR	fO	129	a	Schleifenanfang
GET	gE	161	!	holt ein Zeichen aus dem Tastaturpuffer (falls er nicht leer ist)
GET#	gE#	161&35		empfängt ein Zeichen von externem Gerät
GOSUB	goS	141	m	Sprung mit Gedächtnis zur Rückkehr
GOTO	gO	137	i	Sprung zur angegebenen Zeilennummer
IF	if	139	l	es beginnt Bedingung für THEN
INPUT	input	132	d	(Schreiben eines Textes und) Warten auf eine Eingabe, die der angegebenen Variablen zugewiesen wird
INPUT#	iN	133	e	Empfang einer Zahl oder eines String von einem externen Gerät
LET	let	136	h	es folgt eine Wertzuweisung (dieses Schlüsselwort ist entbehrlich)
LIST	lI	155	[	Auslisten des BASIC-Programms, ganz oder in den gesetzten Grenzen
LOAD	lO	147	s	Einlesen eines Programms von der Tonbandcassette oder der Diskettenstation, zugleich Löschen eines alten Programms im Rechner
NEXT	nE	130	b	Schleifenindex wird um 1 oder den bei STEP angegebenen Wert erhöht, Rücksprung zum Schleifenanfang (FOR)

NEW	new	162	"	Löschen eines BASIC-Programms (betrifft nicht Maschinenprogramme)
NOT	nO	168	(	logische Verneinung des Folgenden
ON	on	145	q	gemäß dem folgenden Wert wird die Sprungadresse für folgendes GOTO oder GOSUB gewählt, z.B. ON X GOTO 1ØØ,2ØØ,3ØØ bewirkt für X=2 Sprung nach 2ØØ
OPEN	oP	159	←	öffnet File für Dialog mit externem Gerät
OR	or	176	Ø	logische Adjunktion (nichtausschließendes ODER)
POKE	pO	151	w	POKE a,b belegt Speicherplatz a mit dem neuen Inhalt b (Ø bis 255)
PRINT	?	153	y	schreibt auf den Bildschirm Inhalt von Variablen oder (mit ") Klartext
PRINT#	pR	152	x	schreibt auf externem Gerät
READ	rE	135	g	Zuweisung aus dem DATA-Paket an Variable
REM	rem	143	o	der Rest der Zeile wird ignoriert
RESTORE	reS	14Ø	l	Zeiger für READ wird auf Anfang des DATA-Stapels gesetzt
RETURN	reT	142	n	Rücksprung hinter letztes GOSUB
RUN	rU	138	j	startet BASIC-Programm
SAVE	sA	148	t	schreibt BASIC-Programm auf Tonband-Cassette oder Diskettenstation
SPC(	sP	166	&	überspringt beim Schreiben Plätze (setzt aber keineswegs Leerzeichen !)
STEP	stE	169	)	Schrittweite für Laufindex in FOR-Schleife (auch negativ)
STOP	sT	144	p	Ende des Laufs mit Meldung der Zeile
SYS	sY	158	↑	Sprung in Maschinenprogramm, das bei dem angegebenen Speicherplatz beginnt
TAB(	tA	163	*	Sprung beim Schreiben in angegebene Spalte (nur falls nach rechts)
THEN	tH	167	'	Sprung zur nächsten Zeile, falls zwischen IF und THEN Ø steht, z.B. nichterfüllte Bedingung, sonst Sprung in Zeile, die hinter THEN steht oder Ausführung aller Anweisungen hinter THEN

TO	to	164	$	steht zwischen Anfangs- und Endwert des Laufindex hinter FOR
VERIFY	vE	149	u	Vergleich, ob Programm im Speicher gleich dem auf Band oder Diskette ist (OK bzw. ERROR)
WAIT	wA	146	r	Unterbrechung des Laufs, bis in einem anzugebenen Speicherplatz eine anzugebende Bedingung eintritt (wird kaum jemals benötigt).

LISTE DER SIMON-SCHLÜSSELWÖRTER

1 HIRES 2 PLOT 3 LINE 4 BLOCK 5 FCHR 6 FCOL
7 FILL 8 REC 9 ROT 10 DRAW 11 CHAR 12 HI COL
13 INV 14 FRAC 15 MOVE 16 PLACE 17 UBB 18 UPW
19 LEFTW 20 LEFTB 21 DOWNB 22 DOWNW 23 RIGHTB
24 RIGHTW 25 MULTI 26 COLOUR 27 MMOB 28 BFLASH
29 MOB SET 30 MUSIC 31 FLASH 32 REPEAT 33 PLAY
35 CENTRE 36 ENVELOPE 37 CGOTO 38 WAVE 39 FETCH
40 AT(41 UNTIL 44 USE 46 GLOBAL 48 RESET 49 PROC
50 CALL 51 EXEC 52 END PROC 53 EXIT 54 END LOOP
58 LOOP 59 DELAY 64 SECURE 65 DISAPA 66 CIRCLE
67 ON ERROR 68 NO ERROR 69 LOCAL 70 RCOMP 71 ELSE
72 RETRACE 73 TRACE 74 DIR 75 PAGE 76 DUMP
77 FIND 78 OPTION 79 AUTO 80 OLD 81 JOY 82 MOD
83 DIV 85 DUP 86 INKEY 87 INST 88 TEST 89 LIN
90 EXOR 91 INSERT 92 POT 93 PENX 95 PENY 98 DESIGN
99 RLOCMOB 100 CMOB 102 PAUSE 104 MOB OFF 10& OFF
106 ANGL 107 ARC 108 COLD 109 SCRSV 110 SCRLD
111 TEXT 112 CSET 113 VOL 114 DISK 115 HRDCPY
116 KEY 117 PAINT 118 LOW COL 119 COPY 120 MERGE
121 RENUMBER 122 MEM 123 DETECT 124 CHECK 125 DISPLAY

Die SIMON-Schlüsselwörter erscheinen im Programmspeicher mit dem Byte 100 und der Zahl, die ihnen in dieser Auflistung vorangeht. Wegen der Bedeutungen vergleichen Sie bitte das Register und das SIMON-Handbuch.

TABELLEN FÜR DIE BENUTZUNG DER FESTEN GRAFIKZEICHEN

Die Zeichen können auf drei Arten angesteuert werden: mit POKE und Bildschirmcode, mit PRINT und ASCII-Code (aufzurufen mit CHR$()) oder durch Schreiben mit der Taste in eine PRINT-Anweisung. Die Taste ist dabei entweder mit SHIFT oder mit der Commodore-Taste C= umzuschalten (Unterstreichung deutet im folgenden das letztere an). Einige Zeichen sind revers, sie haben eine eigene Bildschirmcode-Nummer für POKE, müssen aber bei PRINT mit REVERSE-ON vorbereitet werden. Im folgenden wird das durch Überstreichen angedeutet.

Für die reversen Zeichen ist im Bildschirmcode jeweils 128 zu addieren (bzw. abzuziehen, wenn das Vorbild schon revers ist), bei PRINT ist zwischen REVERSE ON und REVERSE OFF zu wechseln.

1. Waagerechte Linien und Stufen

POKE	100	82	70	64	67	68	69	99
CHR$	164	210	198	192	195	196	197	163
Taste	@	R	P	*	C	D	E	T

POKE	100	111	121	98	248	247	227	160
CHR$	164	175	185	162	184	183	163	32
Taste	@	P	O	I	U	Y	T	leer

POKE	99	119	120	226	249	239	228	160
CHR$	163	183	184	162	185	175	164	32
Taste	T	Y	U	I	O	P	@	leer

2. Senkrechte Linien und Stufen

POKE	CHR$	Taste	POKE	CHR$	Taste	POKE	CHR$	Taste
101	165	G	101	165	G	103	217	M
84	212	T	116	180	H	106	170	N
71	199	G	117	181	J	118	182	L
66	194	B	97	161	K	225	161	K
93	221	-	246	182	L	245	181	J
72	200	H	234	170	N	244	180	H
89	217	Y	231	217	M	229	165	G
103	167	M	160	32	leer	160	32	leer

3. Umrandungsstücke in Buchstabenmitte

POKE	CHR$	Taste
85 73	213 201	U I
74 75	202 203	J K

POKE	CHR$	Taste
112 110	176 174	A S
109 125	173 189	Z X

POKE	CHR$	Taste
113	177	E
115 107	179 171	W Q
114	178	R

POKE	CHR$	Taste
64 91 93	192 219 221	* + -

4. Umrandung am Buchstabenrand

POKE	CHR$	Taste
79 99 80	207 163 208	O T P
101 103	165 167	G M
76 100 122	204 164 250	L @ @

5. Diagonalen

POKE	CHR$	Taste
78 86 77	206 214 205	N V M

6. Pfeile nach 8 Richtungen

	POKE			CHR$			Taste		
	112	30	110	176	94	174	$\underline{A}$	↑	$\underline{S}$
	60		62	60		62	<		>
	109	22	125	173	86	189	$\underline{Z}$	V	$\underline{X}$

7. Bausteine für die Viertelgrafik (vgl.S. 96)

(die Verwendung der hier vorgeschlagenen Indices erlaubt eine sehr rationelle Verknüpfung mit Koordinaten)

Index	0	1	2	3	4	5	6	7
POKE	32	126	124	226	123	97	255	236
CHR$	32	190	188	$\overline{162}$	187	161	$\overline{191}$	$\overline{172}$
Taste	leer	V	$\underline{C}$	$\overline{I}$	$\underline{F}$	K	$\underline{\overline{B}}$	$\overline{D}$

Index	8	9	10	11	12	13	14	15
POKE	108	127	225	251	98	252	254	160
CHR$	172	191	$\overline{161}$	$\overline{187}$	162	$\overline{188}$	$\overline{190}$	$\overline{32}$
Taste	$\underline{D}$	$\underline{B}$	$\overline{K}$	$\underline{\overline{F}}$	I	$\underline{\overline{C}}$	$\underline{\overline{V}}$	$\overline{\text{leer}}$

8. Dreiecke

	POKE		CHR$		Tasten	
	105	95	169	223	£	$\underline{*}$
	223	233	$\overline{223}$	$\overline{169}$	$\underline{\overline{*}}$	$\overline{£}$

9. Halbe Felder

	POKE		CHR$		Tasten	
	226		$\overline{162}$		$\overline{I}$	
	97	225	161	$\overline{161}$	$\underline{K}$	$\underline{\overline{K}}$
	98		162		$\underline{I}$	

10. Ganze Felder

	POKE			CHR$			Tasten		
	32	102	160	32	230	$\overline{32}$	leer	+	$\overline{\text{leer}}$

11. Kreise, Spielkarten etc.

POKE	81	87	83	9Ø	65	88	35	42	43	46
CHR$	2Ø9	215	211	218	193	216	35	42	43	46
Taste	Q	W	S	Z	A	X	#	*	+	.

ZEICHENSÄTZE

```
100 REM ZEICHENSAETZE
110 :
120 HIRES0,1
130 FORI=0TO31:FORJ=0TO15
140 CHAR I*10,J*12,I+32*J,1,1:NEXT:NEXT
150 COPY:REM DRUCKER VC 1525
160 GOTO 160
```

LIST-Zeichen

Die Merkzeichen für Farben, Reverse, Cursor etc. werden in den LISTings mit diesen Zeichen wiedergegeben:

BLK 0 SCHWARZ	8 ORANGE
WHT 1 WEISS	9 BRAUN
RED 2 ROT	10 HELLROT
CYN 3 CYAN	11 DUNKELGRAU
PUR 4 PURPUR	12 MITTELGRAU
GRN 5 GRUEN	13 HELLGRUEN
BLU 6 BLAU	14 HELLBLAU
YEL 7 GELB	15 HELLGRAU

CURSOR:

AUF

LINKS RECHTS

AB

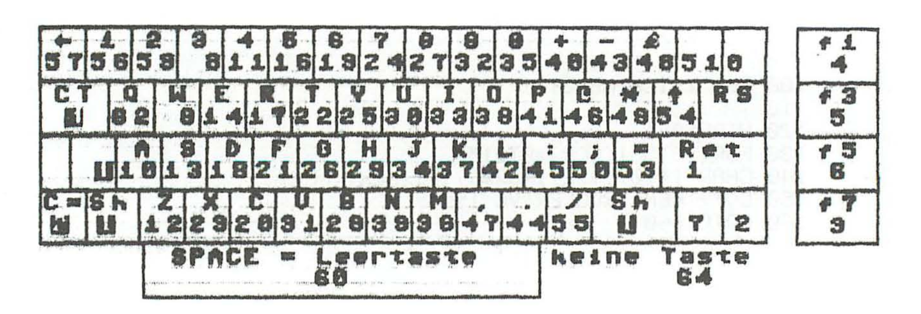
HOME CLEAR HOME

DELETE INSERT

REVERSE EIN (ON) REVERSE AUS (OFF)

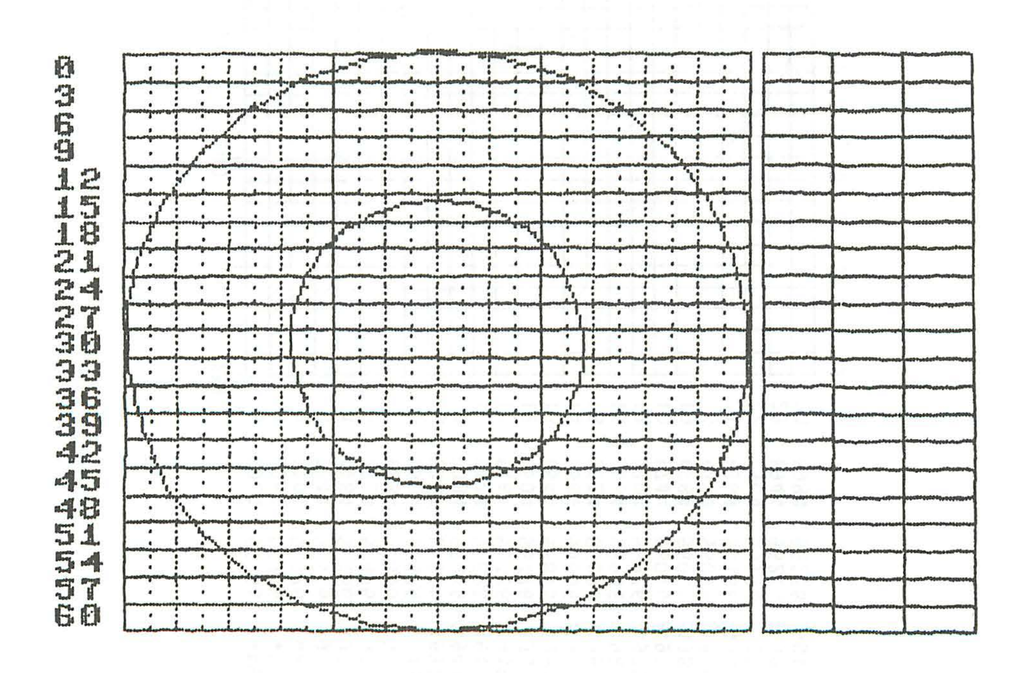
PEEK(203) und PEEK(653)

Zur Abfrage der Tastatur über PEEK(2Ø3) gelten die schwarzen Zahlen, für PEEK(653) die Summe der reversen Zahlen 1 2 4:

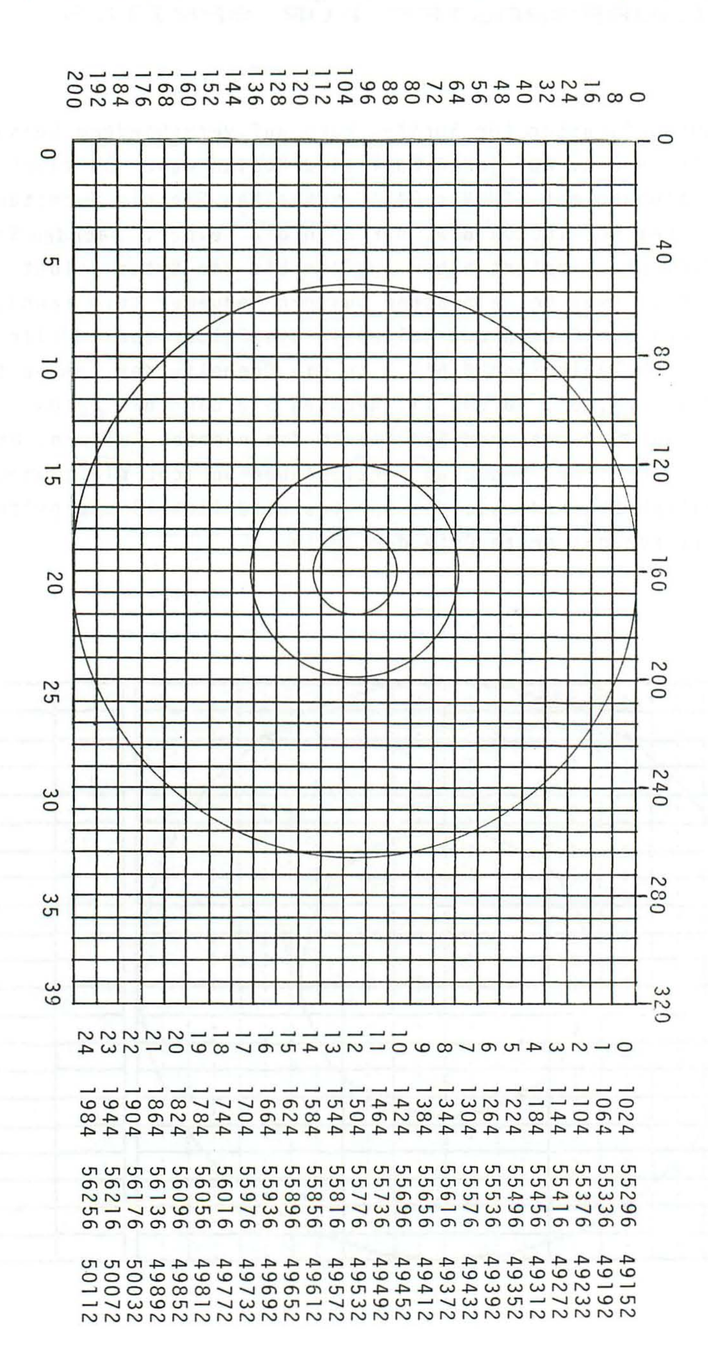


ENTWURFSRASTER FÜR SPRITES

Das Entwurfsraster für Sprites kann auf verschiedene Weisen genutzt werden. Auf jeden Fall sind Kopien davon nützlich. In Verbindung mit SIMON's BASIC schreiben Sie die Buchstaben A B C (bei MULTICOLOR bzw. nur B in die Felder, nachdem Sie die Umrisse skizziert haben. Wollen Sie die Bytes selbst ausrechnen (was bei einfachen Mustern bequemer sein kann), schreiben Sie Einsen und Nullen in die Felder (bei MULTIcolor evtl. auch Zahlen von Ø bis 3 in die Doppelfelder, am besten binär als ØØ, Ø1, 1Ø und 11. Rechnen Sie dann die Bytes daraus aus und notieren Sie sie in den rechten Feldern. Die Zahlen ganz links bedeuten fortlaufende Speicherplatznummer (zuzüglich Startadresse des Sprites natürlich !) und gelten jeweils für das erste Byte der Zeile.



RASTERVORLAGE FÜR GRAFIK



HINWEISE AUF SOFTWARE

Es kann keine Rede davon sein, hier alle Programme zu behandeln, die man für den C 64 kaufen kann: es gibt da ein reichhaltiges, ständig wachsendes Angebot von Cassetten, Disketten, Steckmoduln, aber auch schriftlich erhältlichen Programmen. Das sind Spiele (leider darunter auch viele, die den Spielern sehr destruktive Tätigkeiten zuweisen), Lernprogramme und Hilfsmittel für Textbearbeitung oder Buchhaltungsaufgaben im Beruf. Viele davon sind gegen unberechtigte Vervielfältigung geschützt und damit auch gegen Änderungen, die Sie vielleicht an einem Programm vornehmen möchten, um es noch stärker auf Ihre Bedürfnisse oder Vorlieben hin anzupassen. Auch sind gerade die raffiniertesten Spielprogramme in Maschinensprache geschrieben, um mit hohem Tempo Musik und Bilder zu ermöglichen, und Eingriffe in Maschinenprogramme sind schwieriger als bei BASIC, besonders bei solchen, die man nicht selbst geschrieben hat. Wenn Sie dieses Buch benutzen, ist anzunehmen, daß Sie den C 64 nicht nur dazu benutzen, um fertige Telespiele auf ihm laufen zu lassen (obwohl das auch Spaß macht), sondern daß Sie das Gerät noch kreativer gebrauchen. Dazu braucht man im Prinzip keine fertigen Programme, oder allenfalls um sich Anregungen zu holen über das, was auf der Maschine möglich ist. Es kann aber auch nützlich sein, von fertigen Programmen, die man ausLISTen kann, auszugehen und an ihnen nach eigenem Geschmack weiterzumachen. Dazu ein Beispiel: Das Spiel "TUERME VON HANOI" benutzt die ganz einfache Tastaturgrafik für die Steinchen, beim Bewegen tauschen diese unvermittelt ihre Plätze. Eine reizvolle Variante benutzt Sprites für die Steinchen und bewegt sie auf Wurfparabeln zwischen ihren Positionen.
Auch wenn Sie ein Programm nicht ändern oder ausbauen wollen, ist es vielleicht interessant für Sie, das Innenleben zu durchschauen. In einigen Fällen (der "erklärenden Simulation" in der Physik und anderen Natur-, aber auch etwa Sozial-Wissenschaften) ist gerade der Zusammenhang zwischen der LIST und dem RUN das wichtigste Hilfsmittel zum Verständnis. Ganz am Schluß dieses Buches finden Sie einige Beispiele dafür vermerkt.

LITERATUR-EMPFEHLUNGEN

1. Gebrauchsanleitungen, deren Benutzung parallel zu diesem Buch empfohlen wird:
zum Commodore 64, zu SIMON's BASIC, Zur Diskettenstation VC 1541, zu den Druckern VC 1515, VC 1525 oder MPS 8Ø1, zum Joystick, zu den Paddles, zur Datasette, insebdonsere auch das Commodore 64 reference Guide

2. Bücher speziell zum C 64:

N. Treitz, Spiele mit Computergrafik (mit Programmen, die unmittelbar auf dem C 64 mit SIMON's BASIC laufen) Düsseldorf 1984 (Hagemann)

H. Riedl, F. Quinke, Commodore 64, Ludwigshafen 1983 (Kiehl)

Angerhausen, Becker, Englisch, Gerits, 64 intern, Düsseldorf 1983 (Data Becker) (enthält ROM-Listing)

Angerhausen, Englisch, Gerits, 64 Tips & Tricks, Düsseldorf 1983 (Data Becker)

Plenge, Szczepanowski, Das Trainingsbuch zu SIMON's BASIC, Düsseldorf 1983 (Data Becker)

L. Englisch, Das Maschinensprache Buch zum Commodore 64, Düsseldorf 1984 (Data Becker)

Angerhausen, Becker, Gerits, Schellenberger, 64 für Profis, Düsseldorf 1983 (Data Becker)

R.Babel, M.Krause, A.Dripke, Das Interface Age Systemhandbuch zum Commodore 64 und VC 20, München 1983 (Interface Age)

C. Lorenz, Beherrschen Sie Ihren Commodore 64, Holzkirchen 1983 (Hofacker)

E. Floegel, Hardware-Erweiterungen für Commodore 64, Holzkirchen 1984 (Hofacker)

C. Lorenz, Programmieren in Maschinenspraahe mit dem Commodore 64, Holzkirchen 1984 (Hofacker)

3. Bücher über den mit dem C 64 verwandten VC 2Ø

N. Treitz, Besser programmieren mit dem VC 2Ø, Düsseldorf 1983 (Hagemann); teilweise textidentisch mit dem vorliegenden Buch, evtl. interessant für C-64-Besitzer, die ihren VC 2Ø noch nicht verkauft haben; dort auch weitere Literatur über den VC 2Ø

4. Beschreibung des Commodore-BASIC (auch für den C 64, aber ohne SIMON's BASIC):

H. Schießl, CBM-BASIC-Handbuch, München 1981 (Mediscript)

5. Praktische Einführungen in BASIC (nicht speziell auf den C 64 zugeschnitten, aber auch für ihn geeignet):

Rüdeger Baumann, BASIC Eine Einführung in das Programmieren 198Ø Stuttgart (Klett); dazu auch eine Aufgabensammlung

Rüdeger Baumann, Strukturiertes Programmieren mit BASIC, 1983 Stuttgart (Klett)

Wilfried Schupp, Schüler programmieren in BASIC, Paderborn 198Ø (Schöningh)

6. Sammlungen von Spielprogrammen, die als Ausgangsbasis oder Anregung für C 64-Programme dienen können:

David H. Ahl (ed.) BASIC Computer Games, Morristown (NJ) 1978 (Creative Computing Press)

-, More BASIC Computer Games, Morristown (NJ) 1979 (Cr.C.Pr.)

7. Bücher zur Mathematik, die als Anregung zum Programmieren besonders geeignet sind:

Werner Gilde, Siegfried Altrichter, Mehr Spaß mit dem Taschenrechner, Thun, Frankfurt/M 198Ø (Deutsch)

Roland Stowasser, Benno Mohry, Rekursive Verfahren, Hannover 1978 (Schroedel)

Karl Udo Bromm, Programmierbare Taschenrechner in Schule und Ausbildung, Braunschweig 1979 (Vieweg)

Arthur Engel, Elementarmathematik vom algorithmsichen Standpunkt, Stuttgart 1977 (Klett)

Hans Bartel, Zahlentheorie und (Zahl)zeichensysteme, Würzburg 1976 (Vogel/Kamprath-Reihe)

Hans J. Stetter, Numerik für Informatiker, München 1976 (Oldenbourg)

8. Bücher zum Computereinsatz in Physik und Astronomie

Reinhard Kunze, Rechenprogramme für den Physikunterricht, Köln 1982 (Aulis) (laufen direkt auf CBM 3ØØ1)

Robert M. Eisberg, Mathematische Physik für Benutzer programmierbarer Taschenrechner, München 1978 (Oldenbourg)

G. Venz, Lösung von Differentialgleichungen mit programmierbaren Taschenrechnern, München 1978 (Oldenbourg)

Peter Duffett-Smith, Practical Astronomy with Your Calculator, Cambridge (GB) [2]1981 (Cambridge University Press)

Eric Burgess, Celestial BASIC, Berkeley (Calif.) 1982 (SYBEX)

9. Schulbuch mit mehreren Programmen für den VC 2Ø:

Franz Bader, Friedrich Dorn, Physik Oberstufe MS, Hannover 1983 (Schroedel) (die Programme sind erstmalig in dieser Neubearbeitung enthalten)

10. ein Biophysikbuch, das vielfältige Anregung zum Programmieren bietet:

Manfred Eigen, Ruthild Winkler, Das Spiel, München 1975 (Piper)

11. einige Bücher zu speziellen Themen:

Ryszard Zieliński, Erzeugung von Zufallszahlen, Leipzig 1978 (VEB Fachbuchverlag) und Thun 1978 (Deutsch)

Ludek Pachmann, Vas I. Kühnmund, Computerschach, München 198Ø (Heyne)

Rainer Hahn, Höhere Programmiersprachen im Vergleich, Wiesbaden 1981 (Akademische Verlagsgesellschaft)

Abraham A. Moles, Informationstheorie und ästhetische Wahrnehmung, Köln 1971(DuMont)

Herbert W. Franke, Phänomen Kunst (Die kybernetischen Grundlagen der Ästhetik), Köln 1974 (DuMont)

H. W. Franke, Computergraphik Computerkunst, München 1971 (Bruckmann)

H. W. Franke, Die geheime Nachricht, 1982 Ffm (Umschau)

P. S. Stevens, Zauber der Formen in der Natur, 1983 München (Oldenbourg)

R E G I S T E R
U N D   G L O S S A R

Dieses Register gibt zugleich knappe Worterklärungen oder andere Zusatzinformationen. Die angegebenen Seitenzahlen verweisen auf die jeweils wichtigen Stellen. Oft spielt das Suchwort auch noch auf den weiteren Seiten des gleichen Abschnitts eine wichtige Rolle.

Eingeklämmerte Seitenzahlen bedeuten, daß das Stichwort selbst im Text nicht verwendet wird, aber die Sache bezeichnet.

Beachten Sie bitte diese Abkürzungen, die wesentlich zum Verständnis des Registers nötig sind:

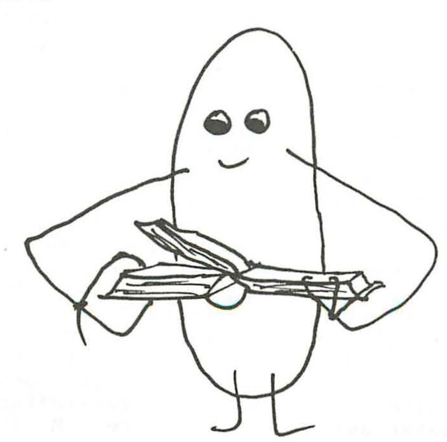
BA Anweisung in Microsoft-BASIC
BF Funktion in Microsoft-BASIC

SA Anweisung in SIMON's BASIC(-Erweiterung)
SAF " in Feingrafik anwendbar
SAG " in grober Grafik (Buchstabenmodus) anwendbar
SF Funktion in SIMON's BASIC
SFF " in Feingrafik anwendbar

T Tastenfunktion
P Programmbeispiel
M Musik und Akustik

Wegen der Syntax der nur im Register, aber sonst nicht in diesem Buch erwähnten Schlüsselwörter vergleichen Sie bitte die zugehörigen Handbücher !

In einigen Fällen haben Anweisungen und Tasten gleiche Namen, obwohl ihre Funktionen völlig verschieden sind (z.B. RESTORE, RETURN, INST)



AUF WIEDERSEHEN !

Programme von N. Treitz für C 64 und VC 20 bei Hagemann:

Die im folgenden genannten Programme und Lehreinheiten sind für den VC 20 und für den C 64 (meist auf dem gleichen Datenträger für beide Rechner in verschiedenen Versionen) erhältlich bzw. sind im Erscheinen. Über Liefermöglichkeiten und den aktuellen Stand der Liste informiert Sie gerne der Verlag Wilhelm Hagemann Karlstraße 20 4000 Düsseldorf.

BESCHLEUNIGUNG Diese Lehreinheit enthält mehrere fertige Programme zur beschleunigten Bewegung in zwei Dimensionen, darunter auch eins über Planetenbahnen. Mit dem Joystick erleben Sie handgreiflich die Folgen der Beschleunigung auf die im Bild ständig gezeigten Positionen Ihres "Fahrzeuges"; Ihre Geschicklichkeit wird bei der Fahrt durch ein Labyrinth gefordert. Das Begleitheft entwickelt die entscheidenden Teile dieser Programme von den ersten Anfängen an, indem der Reihe nach alle physikalisch wichtigen Zeilen erklärt werden. Im Zusammenhang mit dem Programmtext erklären diese Simulationsprogramme dann physikalische Beziehungen zwischen Theorie und Realität.

SCHWINGUNGEN I. Das gleiche gilt auch für diese Lehreinheit: eine Simulation läuft (mit Hilfe der "Uhr" TI zeitgleich mit einem richtigen Federpendel); dabei wird nur ein sehr einfacher Lehrsatz in das Programm "getan": das Gesetz von HOOKE. Mit dem gleichen Gesetz erklärt die nächste Ausbaustufe des Programms dann auch so seltsame Dinge wie "gekoppelte Schwingungen"; ein einfacher Ansatz bewältigt dann die Dämpfung und Resonanzerscheinungen.

SCHWINGUNGEN II. Was FM und AM bedeutet, wird in diesem Programmpaket leicht verständlich, aber auch kompliziertere Dinge werden überschaubar: Pulsfrequenzmodulation, die Codierung von Bild und Ton auf der Bildplatte und die "Zerlegung" und Rückgewinnung von Tönen nach ihrem Klangspektrum mit der FOURIER-Darstellung werden rechnerisch und graphisch übersichtlich geleistet.

WELLENWANNE. Acht Bildschirmbilder, die sich im Takt von 1/16 Sekunde abwechseln, zeigen Wellen und ihre Überlagerungen in zwei Dimensionen, nachdem Sie nähere Einzelheiten vor der

Berechnung der Bilder festgelegt haben. Man kann sich anhand des Programmtextes davon überzeugen, daß eine Überlagerung wirklich nichts anderes ist als eine punktweise Addition.

LACKMUS. Chemische Reaktionen werden in einer Schleife simuliert, die Konzentrationen laufend grafisch dargestellt. Das Massenwirkungsgesetz und die Reaktionsgleichung bestimmen den Ablauf, auch bei willkürlichen Eingriffen in die Konzentration (simuliert mit den Tasten).

TUERME VON HANOI. Ein Stapel von Scheibchen muß nach einer einfachen Spielregel verlagert werden. Das Programm verhindert das Mogeln und teilt am Schluß mit, ob die Züge optimal waren oder nicht.

INTERFERENZ AN SPALTEN. Sie geben eine Anordnung von mehreren Spalten (Zahl, Breite, Abstand) ein. Das Programm bestimmt dann die Interferenzerscheinung (wahlweise als Intensität oder Amplituden dargestellt), wobei es während der Rechnung die dazu benutzten Zeigerdiagramme zeichnet.

STRAHLENOPTIK. Wie sehen die Strahlengänge ohne die üblichen Idealisierungen aus ? Wie arbeiten Telekonverter und ZOOM ? Die Programme für Hohlspiegel und Linsensystem zeigen es, natürlich in Feingrafik. Sie können aber auch die Idealisierung für achsennahe Bündel einschalten und trotzdem in gedehntem Maßstab zeichnen lassen.

FARBENSEHEN. Der Zusammenhang zwischen Lichtspektrum, Reizung der Sehzellen der Netzhaut und Farbempfindung ist nicht ganz einfach. Drei Programme ersetzen komplizierte Mathematik durch einfache grafische Operationen auf dem Bildschirm und vereinfachen dabei mehr oder weniger stark quantitativ.

MUSIK: QUINTENZIRKEL und ALEATORIK. Melodien oder Akkorde, die Sie direkt auf der Tastatur (die wie ein Klavier bzw. wie das linke Manual eines Akkordeons abgefragt wird) spielen, erklingen nicht nur, sondern werden auch durch Farben und Positionen grafisch angezeigt. Bei der ALEATORIK können Sie während der Zufallsmusik Tonalität, Stärke des Wiederholungsanteils und Intervallgröße verändern.

THERMOSTAT. Eine abgemagerte Fassung dieses Programms finden Sie in diesem Buch auf Seite 223.

ZUFALL. Dieses Programmpaket enthält unter anderem die grafische Simulation eines GALTON-Brettes, bei dem Kugeln durch mehrere Reihen Nägel fallen.

ELEMENTE. Das Periodensystem wird auf den Bildschirm geschrieben. Vorher haben Sie aber Kriterien und Schwellenwerte eingegeben, so daß an der Färbung z.B. abgelesen werden kann, welche Elemente eine bestimmte Dichte überschreiten, eine bestimmte Elektronegativität haben etc.etc.

BEZUGSSYSTEME. Gleichzeitig wird ein Vorgang auf dem Schirm nebeneinander in zwei verschiedenen Bezugssystemen dargestellt, z.B. einem rotierenden oder einem Schwerpunktssystem usw.

ELEKTRISCHE FELDER. Sie geben einige Punkte mit ihren Koordinaten und Ladungen vor; auf dem Bildschirm werden dann Potentiale, Feldstärkebeträge und/oder Feldstärke-Richtungen durch Färbungen und Pfeile dargestellt.

FRANZÖSISCHE VOKABELN. Ein Vokabelquiz mit allen Akzenten, Umlauten usw. Sie können den Wortschatz auch selbst ergänzen oder ändern.

GESICHTER. Dieses Scherzprogramm zeichnet Gesichter aufgrund von wenigen Bits und demonstriert dadurch einen Zusammenhang zwischen Bild und Information, der weitaus rationeller als ein Raster ist.

SPRACHSCHERZE. Tiernamen oder Soziologenchinesisch, mit dem Zufallsgenerator aus wenigen Wortelementen erzeugt, und durchgeschüttelte Sprichwörter oder Zitate erzeugen eine Komik, die der Computer ganz unschuldig liefert.

MEMORY/WURM/WUERMER. Drei Spiele: Memory mit Bildern und zwei Versionen des gefräßigen Wurms für einen oder für mehrere Spieler.

SENSO/BARRIERE/UHR/WANDERER. Farben und Töne wahrnehmen, behalten und eintippen, ist nicht schwer, wird es aber, wenn die zufällig erzeugte Kette immer länger wird. BARRIERE ist ein Spiel, bei dem es auf Reaktion und Strategie ankommt, die UHR und der WANDERER sind kleine grafische Beigaben.

Spiele mit Computer-Grafik (!)

Einfache und wirkungsvolle Grafikprogramme
für den Commodore 64 mit SIMON's BASIC, auch auf andere Rechner übertragbar

- zum Anschauen und Genießen, Eintippen und Abwandeln
- Zufallsbilder, optische Täuschungen, Beispiele aus Mathematik, Physik,Geographie,
- vielfache Effekte: Symmetrien, Drehungen, Verzerrungen, Verwandlungen, Perspektive, Stereo, Verzweigungen usw.

Zu allen Programmen gibt es z.T. mehrfarbige Bilder, die vollständige (meist sehr kurze !) AusLISTung der C-64-Version und einen Kommentar zum Hintergrund mit Anregungen zu Variationen. Das Buch zeigt auf ästhetisch eindrucksvolle Weise, welche Möglichkeiten die Grafik auf preiswerten Bildschirmrechnern wie dem C 64 bietet und wie man sie nutzen kann. Es lädt durch die Beispiele nicht nur zum Abändern, sondern auch zum eigenständigen Kombinieren der Grundideen ein und fordert daher die Kreativität des Lesers (falls er einen Computer zur Verfügung hat) heraus.

Norbert Treitz, Spiele mit Computer-Grafik (!)
HADÜ-Hagemann, Lehrmittel- und Verlagsgesellschaft mbH.
Düsseldorf 1984, 128 Seiten, davon 12 mehrfarbig

Physikdidaktische Veröffentlichungen von N. Treitz über "erklärende" Simulationen mit ausgedruckten Programmbeispielen finden Sie in den Zeitschriften:

Praxis der Naturwissenschaften/Physik 31 (1982),173-181
LOG IN 3 (1983) 31-39
Der Physikunterricht 2/1983, 6-16
Naturwissenschaften im Unterricht/Physik/Chemie 12 (1983) Heft 12 und folgende Hefte (Serie)
LOG IN 1 (1984) und folgende Hefte (Serie)

Mehr als nur ein Basic-Kurs, denn Sie finden in diesem Buch

- knappe Beschreibungen, kurze Beispiele und Tricks zu den wirklich wichtigen BASIC-Anweisungen - auch zu solchen der Erweiterung "SIMON's BASIC"
- Eingaben mit INPUT, GET, PEEK, Paddles und Joystick
- Editiertricks für Faulenzer
- was man mit dem Zufallsgenerator machen kann
- viele Sorten Grob- und Feingrafik, ein- und mehrfarbig, mit und ohne SIMON's BASIC, auch 64000 Punkte mit jeweils 16 Vorder- und 16 Hintergrundfarben in 1000 Segmenten
- Verwendung und Erzeugung von SPRITEs; auch wie der Rechner Ihre Sprites dreht oder verformt
- Abspeichern von Feingrafik; Zeichnen von Feingrafikbildern mit dem Drucker VC 1525 oder 1515, auch mit mehr Punkten als bei der Hardcopy vom Bildschirm
- einen SYNTHESIZER, verständlich und knapp in BASIC geschrieben, zum Orgelspielen und zum Erklären der vielfältigen Klangmöglichkeiten des C 64: zwei Tastenreihen bilden die Klaviatur, mit den anderen können Sie zwischendurch alle Klangregister verstellen. Die Diskussion der Einzelheiten dieses Programms gibt Ihnen das Verständnis und viele Anregungen für eigene Musik- und Geräuschprogramme
- Anregungen, Tips und kurze Beispiele (die Sie noch ausbauen sollten !) für Simulations- und Spielprogramme für Schule, Selbstunterricht und zum Vergnügen
- Entwurfsraster für Bildschirm und Sprites, auch für den 16 x 16-Farben-Modus bei Hochauflösung
- systematische Tabellen zur sinnvollen Nutzung der auf den Tasten direkt verfügbaren Grafikzeichen mit ihren Nummern für POKE, CHR$ und ihren Zuordnungen zu den Tasten (z.B. auf welcher Taste findet man den senkrechten Strich in der Mitte ?)
- ausführliches Register, zugleich als Kurz-Lexikon

Wenn Sie einen C 64 haben oder kaufen wollen, ist dieses Buch eine anregende, leichtverständliche und preiswerte Ergänzung zum "Commodore 64 MicroComputer Handbuch". Es enthält keine langen Programme zum Eintippen (abgesehen vom SYNTHESIZER), sondern nur kurze Beispiele zum Kennenlernen der Tricks, denn so richtig Spaß macht der Computer erst beim Schreiben von eigenen Programmen und beim Spielen damit. Hier liefert Ihnen dieses Buch Ideen und zeigt Möglichkeiten auf.

ISBN 3-544-53002-3
Best.-Nr. 053002